ANALISIS DE ESTABILIDAD EN PROTOCOLO DE CONCENSO EN ROBOTS NO HOLONÓMICOS EN PRESENCIA DE RETARDOS BAJO EL ALGORITMO DE AGRUPAMIENTO DE RAICES CARACTERÍSTICAS

LUIS FERNANDO PERICO REMOLINA

UNVERSIDAD DE PAMPLONA
FACULTAD DE INGENIERÍAS
PAMPLONA, NORTE DE SANTANDER
2018

ANALISIS DE ESTABILIDAD EN PROTOCOLO DE CONCENSO EN ROBOTS NO HOLONÓMICOS EN PRESENCIA DE RETARDOS BAJO EL ALGORITMO DE AGRUPAMIENTO DE RAICES CARACTERÍSTICAS

LUIS FERNANDO PERICO REMOLINA

TESIS PARA OPTAR POR EL TÍTULO DE MAGISTER EN CONTROLES INDUSTRIALES

M.Sc. JORGE LUIS DIAZ RODRIGUEZ
DIRECTOR

PhD. ALDO PARDO GARCÍA
CODIRECTOR

UNVERSIDAD DE PAMPLONA
FACULTAD DE INGENIERÍAS
PAMPLONA, NORTE DE SANTANDER
2018

AGRADECIMIENTOS

Durante esta etapa de crecimiento han existido muchas personas que se han convertido en el motor para seguir adelante. Agradezco a mis hijos por convertirse en el motivo para seguir. Particularmente le agradezco a Andrea por convertirse en mi soporte de vida tanto en lo académico como en lo emocional, en mi proyecto de vida, sin cuyo concurso ningún proyecto actual estaría en proceso.

Agradezco al doctor Rudy Cepeda por proponer este trabajo de investigación y dar las pautas para realizarlo. Al profesor Jorge Luis Diaz por ser un facilitador de todos los procesos de la maestría desde el inicio y convertirse en un amigo y asesor académico de la Universidad donde laboro.

Contenido

Indice de Figuras

FORMULACIÓN DEL PROBLEMA

Los trabajos en robótica colaborativa han tenido una relevancia grande en los últimos tiempos, ya que permiten realizar trabajos mucho más eficientemente, llámese por costo computacional o por costo de gasto de energía, inclusive por temas económicos dada la construcción más simple de varios robots replicados que uno mucho más complejo.

Uno de los grandes problemas cuando se establece un grupo de robots en entorno colaborativo, llamados enjambres, es la formación o control en posicionamiento. Esto es lograr un consenso entre los miembros del grupo con el fin de que “negocien” la posición que deben cumplir cada uno para mantener la formación. Un protocolo de estas características incluye la comunicación de las posiciones y velocidades de cada uno distribuidas en el grupo. Esta comunicación sufre de un inherente retardo en comunicaciones unido al retardo de cómputo de la acción a emprender, lo que causa una inestabilidad en el sistema que depende del tiempo de dichos retardos. Este trabajo busca validar el trabajo realizado por el Doctor Cepeda en su tesis doctoral que incluye la creación de una ley de control proporcional y derivativa que logre el posicionamiento de cada robot para lograr la formación definida inicialmente que es definida en un grafo de posición.

La idea de este trabajo es lograr calcular previo a la ejecución de la ley de control, los rangos de tiempo de cada retardo que teóricamente hacen que el modelo sea estable, esto es, logren la formación de agentes. Dicho cálculo se realiza con el protocolo de determinación de raíces características (CTCR) que fue planteado inicialmente por el Doctor Olgac en la Universidad de Connecticut en el 2005.

1. JUSTIFICACIÓN

Encontrar los límites de estabilidad en un sistema real que está en presencia de retardos es vital para los sistemas cooperativos modernos, ya que una tendencia en la robótica es hacer elementos menos poderosos mecánicamente pero que sean capaces de realizar labores en conjunto, tal es el caso de los drones para tareas específicas. Los retardos que se presentan en los modelos reales, especialmente en los que aparecen en las comunicaciones, que se pueden reducir pero nunca eliminar, hacen que los sistemas permanezcan estables solo dentro de algunos límites de tiempo. Por eso es muy importante poder calcular dichos límites de estabilidad para poder diseñar y predecir el comportamiento de sistemas robustos que puedan responder a las necesidades de robótica colaborativa moderna. También es importante buscar métodos alternativos que permitan analizar la estabilidad en presencia de retardos para amplios rangos del dominio del tiempo que se conviertan en procedimientos fiables y de una relativa sencilla implementación computacional.

2. OBJETIVOS

OBJETIVO GENERAL

Desarrollar un sistema de control de la posición en un grupo de robots no holonómicos que están bajo la acción de retardos utilizando el algoritmo de agrupamiento de raíces características

OBJETIVOS ESPECÍFICOS

Desarrollar un modelo matemático del algoritmo de consenso que permita evaluar la estabilidad en la formación de agentes cuando se presentan retardos en el sistema

Aplicar el protocolo de las raíces características que permitan establecer las regiones evaluadas en el dominio del tiempo de los retardos presentes en el sistema que conducen a estabilidad

Utilizar la plataforma computacional Matlab que permita realizar la simulación de formación de robots para evaluar el criterio de estabilidad según CTCR

3. ESTADO DEL ARTE

Los sistemas de control en presencia de retardos se han estudiado desde hace décadas motivado por el aspecto de aplicación en sistemas modernos. Muchos procesos de control incluyen fenómenos de tiempos muertos en su dinámica en áreas como la biología química, economía, así como en modelos poblacionales. En los procesos industriales, el tiempo de procesamiento y de propagación en actuadores y sensores es parte de interés especial de los diseñadores de estos sistemas. Al estar alejados unos sensores de los actuadores los retrasos en comunicaciones pueden ser la diferencia en la estabilidad o la pérdida de control del sistema.

En el caso de retardo constante, muchas técnicas diferentes conducen a algoritmos eficientes para probar la estabilidad del sistema de retardo en el dominio del tiempo. Dentro de estas está la teoría de Lyapunov,[1] tal vez el más estudiado de los métodos.
En este último enfoque, tenemos como objetivo encontrar una función de Lyapunov que describa el estado completo del sistema $x_t(\theta)$ el cual no es sencillo de solucionar para un sistema de retraso lineal con un retardo. De hecho para un sistema de retraso lineal la función que define la solución es de muy difícil manipulación. Esta es la razón por la cual la función de Lyapunov-Krasovski (LKF) es más usada por la factibilidad en su implementación. Esta aproximación tiene dos dificultades. La primera es la escogencia del modelo de transformación . La segunda es los límites de algunos términos cuando aparece la derivada en la función de Lyapunov. Se han usado dos técnicas que han probado su efectividad . La primera es adoptar un esquema discreto en forma de las matrices de Lyaponov-Krasovski. Asumiendo el precio de incrementar el número de variables a ser optimizadas, el resultado tiende a ser satisfactorio y convergente.
Otra aproximación interesante usa vector de estado para construir un nuevo sistema alternativo, donde la parte del retardo es desarrollada para construir la función de LK la cual depende de una versión discretizada de todo el estado $x_t(\theta)$. En el caso de retardos variantes en el tiempo los resultados son mucho más pobres y se proponen metodologías aún más conservadoras en el desarrollo matemático, esto es, mucho más complejas en su desarrollo teórico.
La solución clásica que se ha realizado con el teorema de Lyapunov-Krasovskii [1] se obtienen funciones continuas y acotadas que para este caso sería en intervalos de tiempo, para así de esta manera poder analizar si el sistema es estable o no. El análisis de estabilidad es realizado por medio de la derivada de estas funciones continuas, cuya derivada debe ser menor a un valor que se es planteado en estos teoremas, cabe recordar que todo parte del análisis de Lyapunov donde se plantea una ecuación que determine el sistema y su derivada sebe ser decreciente hacia un punto donde se generara estabilidad.

Vale la pena anotar que este análisis es acotado en el tiempo, es decir, se realiza el estudio de estabilidad para un periodo de tiempo establecido, entonces no se encontrarán todos los rangos de valores del retardo que hacen estable al sistema. Para poder tener un análisis de estabilidad en un rango amplio del dominio del tiempo, sería necesario implementar una serie de intervalos en

donde de cada uno se obtendría una función de Lyapunov volviendo el proceso bastante engorroso.
Ahora si el sistema dinámico tiene múltiples retardos, el gasto computacional para el análisis sería mucho mayor y la implementación de los teoremas frecuentemente usados sería más tediosa y seguiría siendo un análisis parcial en el dominio del tiempo de retardos. Aun así se han usado estos teoremas en recientes investigaciones como se puede observar en [2], donde Qingdong Li et al [2], realizan el análisis de control con un tiempo de retardo de un sistema multi-agente con topología de comunicación directa aplicando el anterior nombrado teorema, además del planteamiento de un ejemplo de su respectiva aplicación para su comprobación.

En el trabajo de Yassine Ariba [10] se desarrolla un nuevo tipo de función LF fraccionando el retardo para tomar todo el estado del sistema. Esta idea se complica mucho más si el retardo es variante en el tiempo. Se ha usado entonces aumentar las variables de estado que permite mejor manipulación pero generalmente no da una fiel descripción del sistema. Por esto es que tradicionalmente en la literatura se encuentra que usan unas variables “falsas” para construir artificialmente relaciones lineales entre los argumentos de estado y el estado original del sistema. Estos autores entonces proponen adicionar unas restricciones integrales cuadráticas que mejoran la relación entre la matriz de estado aumentada y el estado original con retrasos

Algunos investigadores han planteado otros métodos de análisis, como el de Zhaoxia Peng et al [3], el cual plantea un protocolo de consenso [20] para un sistema multi-agente heterogéneo donde realizan el análisis con tiempo de retardo por medio del criterio de estabilidad de Nyquist. Por medio de este criterio el autor encuentra el tiempo de retardo mínimo permisible por el sistema.
Derong Liu et al [4], en cambio llegan a hacer uso de la inteligencia artificial específicamente de redes neuronales, ya que parten del hecho de que no conocen el sistema dinámico y asumen que es no lineal; además hacen uso nuevamente de las funciones de Lyapunov-Krasovskii para eliminar los efectos negativos del tiempo de retardo

Con el enfoque basado en el consenso y su estabilidad, algunos investigadores han optado por aplicar control óptimo en sus sistemas multi-agentes, como puede ser observado en [5], donde Ming Xin y Jianan Wang realizan un control óptimo de rastreo distribuido, cuyo control se basa en encontrar una función de costo en la que se disminuyen esfuerzos, se buscan trayectorias y se superan obstáculos en la misma.

Otra reciente investigación fue realizada por Shichun Yang et al [6], basada en un control adaptativo distribuido en robots con restricciones no holonómicas a partir del enfoque de análisis de estabilidad de Lyapunov, todo esto a partir de una transformación de variable.

El protocolo CTCR es desarrollado por Cepeda [11] dando las bases matemáticas para calcular las hipersuperficies que denotan las regiones de estabilidad en presencia de dos retardos.

5. MARCO CONCEPTUAL

5.1 CINEMÁTICA DE UN ROBOT DIFERENCIAL NO HOLÓNOMICO

El trabajo realizado por López [9] realiza un análisis de la cinemática de un robot uniciclo el cual es no holonómico. Un diagrama esquemático de este tipo de robot se muestra en la Figura 1 en la que además se definen las variables que describen su comportamiento.
Debido a que la rueda gira sin deslizar, el uniciclo no puede desplazarse en dirección perpendicular a su velocidad, esta es una restricción no holónoma que puede expresarse mediante la ecuación:

$$\dot{x}\sin\theta - \dot{y}\cos\theta = 0 \qquad (5.1)$$

.

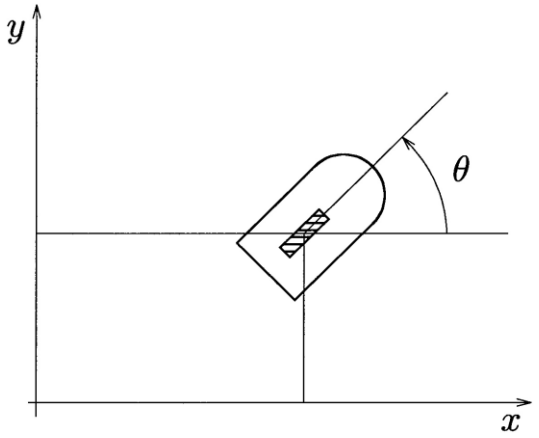


Figura 1. Variables que describen el estado de un uniciclo

Las velocidades posibles del uniciclo, y que pueden ser usadas como entradas de control, son entonces la velocidad lineal $v = \sqrt{\dot{x}^2 + \dot{y}^2}$ y la velocidad angular $\omega = \dot{\theta}$. De esta manera, el modelo cinemático del uniciclo puede expresarse así:

$$\begin{bmatrix}\dot{x}\\ \dot{y}\\ \dot{\theta}\end{bmatrix} = \begin{bmatrix}\cos\theta & 0\\ \sin\theta & 0\\ 0 & 1\end{bmatrix}\begin{bmatrix}v\\ \omega\end{bmatrix} \qquad (5.2)$$

Un robot diferencial tiene una cinemática similar a la del uniciclo. Esta clase de robots cuenta con dos ruedas montadas sobre un mismo eje que pueden ser actuadas de manera independiente. Las entradas de control son las velocidades lineales de cada una de las ruedas. A continuación se analiza la cinemática de un robot diferencial, para compararla con la de un uniciclo.

El centro instantáneo de rotación, el punto alrededor del cual está girando el robot en un instante dado, se encuentra en un punto sobre la proyección del eje sobre el cual las ruedas están montadas, a una distancia r del centro del robot, como se observa en la Figura 2. Puesto que la velocidad angular de todos los puntos sobre el robot es la misma, se pueden obtener las siguientes expresiones:

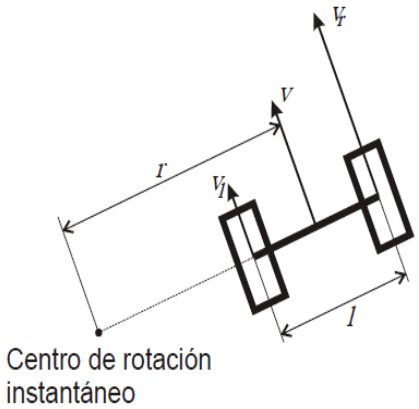


Figura 2. *Velocidades de un robot diferencial*

$$\omega = \frac{v}{r} \qquad (5.3a)$$

$$\omega = \frac{v_r}{r+\frac{l}{2}} \qquad (5.3b)$$

Siendo v_r y v_l las velocidades lineales de las ruedas derecha e izquierda y l la longitud del eje del robot, es decir, la distancia que separa las ruedas. De esta forma se llega a:

$$v_r = \omega\left(r + \frac{l}{2}\right) \qquad (5.4a)$$

$$v_l = \omega\left(r - \frac{l}{2}\right) \qquad (5.4b)$$

Que al sumarlas producen la relación entre la velocidad lineal del robot y las velocidades individuales de las ruedas:

$$v = \frac{v_r + v_l}{2} \qquad (5.5)$$

A partir de (4.3) y (4.5) se obtienen las relaciones directa e inversa entre las velocidades de las ruedas y las velocidades del robot:

$$\begin{bmatrix} v \\ \omega \end{bmatrix} = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} \\ \frac{1}{l} & -\frac{1}{l} \end{bmatrix} \begin{bmatrix} v_r \\ v_l \end{bmatrix} \quad (5.6a)$$

$$\begin{bmatrix} v_r \\ v_l \end{bmatrix} = \begin{bmatrix} 1 & \frac{l}{2} \\ 1 & -\frac{l}{2} \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix} \quad (5.6b)$$

Estas fueron las ecuaciones que López trabajó en su proyecto de grado en la programación del microcontrolador con el cual logró la convergencia de los robots, haciendo que uno de ellos recibiera coordenadas absolutas de destino inalámbricamente, y el segundo robot recibía permanentemente las coordenadas del segundo, haciendo que los dos se dirigieran al mismo punto en formación.

5.2. Espacio de estados

En sistemas de control clásico es usado un método de análisis basado en función de transferencia. Para situaciones de complejidad de análisis elevado como sistemas MIMO, se usa un control moderno basado en múltiples ecuaciones diferenciales de primer orden, representadas mediante matrices, que representan también sistemas de primer orden sin altear el sistema; esto permite obtener una representación matemática más sencilla y así un análisis menos tedioso. En la Ecuacion1 se muestra la ecuación de estado de un sistema dinámico:

$$\dot{\mathbf{X}}(t) = \mathbf{A}(t)\mathbf{X}(t) + \mathbf{B}(t)\mathbf{U}(t) \quad (5.7)$$

Donde **A** es la matriz de estado, **B** matriz de entrada, **U** vector de entrada y **X** el vector de estados. [9]

5.3. Protocolo de consenso

Se consideran un grupo de "*n*" agentes que se están comunicando entre ellos mismos. Según su topología de comunicación cada agente que llamaremos "*j*" tendrá cierta cantidad de informadores o un subconjunto de agentes, los cuales informarán su posición y velocidad a "*j*". El consenso [23] es alcanzado si $\lim_{t\to\infty} x_j(t) - x_k(t) = 0$ para todo *k* que pertenezca al subconjunto de agentes con los que se está comunicando y *j* es el agente que se está analizando. Como particularidad los canales de comunicación entre los agentes tendrán un retardo de tiempo medido en segundos. [11][14]

Un método de realizar el control de un protocolo de consenso es calcular el promedio de la posición del grupo de informadores que obtiene cada agente, y a éste restarle su posición; realizando lo mismo para la información de velocidad que se obtenga. Seguidamente se multiplicarán estos resultados por una ganancia, que en el caso de la posición corresponderá a un control proporcional y para la velocidad a uno derivativo.

Es claro que existen muchos métodos de realizar un protocolo de consenso, pero algo que tienen en común todos estos métodos es el hecho en que se incurrirá en un retardo en la comunicación entre los agentes, dado que la transmisión de información nunca es instantánea. Esta situación se verá reflejada en el consenso ya que si este retardo es muy largo, no habrá un acuerdo para llegar a un punto común entre ellos. [16]

5.4. Teoría de grafos

Para representar una red comunicación es útil usar la teoría de grafos. Un grafo $\boldsymbol{\Gamma}$ consiste en un grupo de vértices, de aristas y una relación de incidencia; donde la relación de incidencia relaciona dos vértices con una arista. Cuando un par de vértices tienen una dirección el grafo es llamado directo y si estos no tienen una dirección específica el grafo es llamado indirecto, que también puede llegar a ser directo debido a que hay dirección en los dos sentidos. Otro aspecto importante es el hecho de que cada vértice tiene un grado de salida y de entrada asociado, cuyo grado de entrada hace referencia a la cantidad de vértices que junto con el forman un borde y tienen dirección hacia el mismo; y el grado de salida hace referencia a la cantidad de vértices que junto con el forman un borde y la dirección va hacia esos vértices y no hacia el mismo. En los grafos indirectos el grado de entrada es igual al grado de salida.

La matriz de adyacencia $\mathbf{A}_{\boldsymbol{\Gamma}}$ de un grafo es una matriz cuadrada de $n \times n$, donde n es la cantidad de vértices de la red de comunicación, cuyos coeficientes $a_{jk} \neq 0$ y hacen referencia si hay comunicación o no, con cada k vértice; cuyo valor puede ser escogido por el usuario, como ejemplo podría ser tomado $a_{jk} = 1$. Los coeficientes $a_{jj} = 0$, porque esa posición de la matriz hace referencia a que la comunicación de ese vértice se está haciendo con el mismo.

La matriz grado Δ, es una matriz es una matriz cuadrada de $n \times n$, donde n es la cantidad de vértices de la red de comunicación cuyos términos Δ_{jj} son correspondientes al grado de dicho vértice y los términos $\Delta_{jk} = 0$. [11]

5.5 Método de CTCR (Cluster Treatment of Characteristic Roots)

Partiendo de un sistema dinámico de con múltiples tiempos de retardo:

$$\dot{\mathbf{X}}(t) = \mathbf{A}\mathbf{X}(t) + \sum_{j=1}^{p} \mathbf{B}_j \mathbf{X}_j(t - \tau_p) \qquad (5.8)$$

Se puede obtener la ecuación característica que es la encargada de determinar los polos del sistema dinámico de la siguiente manera incluyendo los tiempos de retardo:

$$CE\left(s, \tau_1, \dots, \tau_p\right) = \det\left[s\mathbf{I} - \mathbf{A} - \sum_{j=1}^{p} \mathbf{B}_j \boldsymbol{e}^{-\tau_j s}\right] \qquad (5.9)$$

El método de CTCR (Cluster Treatment of Characteristic Roots) [12], se basa en identificar los intervalos de tiempo de retardo en los que las raíces de la ecuación característica, se encuentran ubicadas en la frecuencia $s = \omega i$ del plano imaginario; a partir de esto se plantea el uso de la tendencia de raíz, cuya función principal es conocer hacia qué parte del plano imaginario se desplazará la raíz que se encuentra en el eje imaginario, teniendo en cuenta que si se desplaza hasta el semiplano derecho se desestabilizará el sistema, y si se desplaza hasta el semiplano izquierdo será estable. Plasmando lo anterior mencionado a través de un método gráfico, es posible encontrar las áreas donde el sistema es estable con sus respectivos intervalos de tiempo de retardo. La principal ventaja de este método, es que permite identificar todas las áreas de estabilidad sin el uso de las técnicas comunes encontradas en la literatura como lo era acotar cierta área y realizar su respectivo análisis.

La construcción del método de CTCR puede ser resumido en dos pasos. El primer paso es la construcción de las *hipersuperficies kernel*, la cual es una gráfica que representa todas las raíces que se encuentran sobre el eje imaginario con el tiempo más pequeño de retardo. Para el segundo paso hay que tener en cuenta un concepto; donde una raíz de la ecuación característica sea completamente imaginaria, es decir que se encuentre sobre el eje imaginario, generará un sistema con respuesta periódica por lo tanto si en cierto valor de tiempo de retardo se encuentra una raíz completamente imaginaria, es posible decir que si a ese tiempo de retardo le sumamos $2\pi/\omega$, donde ω es la frecuencia donde se genera esta raíz imaginaria, podría tener otra raíz de este tipo en este nuevo tiempo de retardo:

$$< \tau + 2\pi/\omega > \qquad (5.10)$$

En este orden de ideas el segundo paso del método de CTCR es obtener las demás gráficas que surgen de lo periódicas que son estas raíces imaginarias y así posteriormente usando la tendencia de raíz analizar qué área, de la unión de estas gráficas, hace estable el sistema dinámico.

Para sistemas con múltiples tiempos de retardos su respectivo análisis robusto de estabilidad es demasiado complejo, por ello también es posible usar el método de CTCR. Para su implementación es necesario realizar una construcción de bloques en el dominio del tiempo de retardo y bloques de longitud de 2π. Este mapeo por construcción de bloques convierte espectros sistemas de dimensión infinita en espectros de dimensión finita [13].

I. CAPITULO 1

6. METODOLOGÍA

Para realizar el análisis y aplicación del método CTCR a robots cooperativos no holonómicos, se plantearon cinco etapas que buscaron ir entrelazando resultados y conclusiones para cumplir los objetivos planteados. En forma general se describen a continuación los procesos y la metodología usada para el desarrollo de la investigación:

6.1. DOCUMENTACION Y REVISION BIBLIOGRÁFICA

Basados en trabajos realizados en el grupo de investigación de la Universidad Santo Tomás, se construyó el modelo de control basado en elemento proporcionales para la posición y derivativos para la velocidad. La técnica escogida es la linealización por retroalimentación donde se realiza una aproximación a un modelo lineal partiendo de un sistema dinámico de segundo orden.

Basados en las teorías trabajadas por el doctor Olgac se establecieron los principios del protocolo CTCR para encontrar regiones de estabilidad en presencia de retardos

Las bases matemáticas de dicho protocolo se basaron en la revisión de trabajos que expliquen las transformaciones de Rekasius, dada que no se trata de una aproximación si no un método exacto. Incluye la base matemática el uso de la matriz de Silvester que permite la implementación en una herramienta computacional

La implementación del modelamiento de un robot no holonómico estuvo basado en el trabajo de Oriolo et al[15] que plantea el uso de una entrada de segundo orden para cumplir con el protocolo de consenso

6.2. ESTABLECER UNA CONSIGNA DE CONTROL BAJO COMPONENTES PROPORCIONALES Y DERIVATIVOS QUE PERMITAN MODELAR EL SISTEMA CON RETARDOS EN LAS COMUNICACIONES.

El sistema de robot no holonómico es controlado por un modelo de retroalimentación aplicando elementos proporcionales para la posición y derivativos para la velocidad escogidos para poder aplicar el método de CTCR. No es usada otra técnica de control debido a la complejidad matemática ya entablada en este proyecto por eso se lleva a usar una técnica sencilla de control. Esta técnica permite al usuario adaptarse a cualquier sistema de acuerdo con su aplicación. Por ejemplo las restricciones de velocidad de un robot móvil podrán limitar las ganancias para evitar acciones de control mayores a las que él es capaz de ejecutar. Esto también permite comprobar lo exacto y exhaustivo que

puede ser el método de CTCR, ya que demuestra que el análisis puede ser realizado con cualquier tipo de ganancia dada por el usuario evidenciando su adaptabilidad. Lo que se buscó con este proyecto es usar un nuevo método de análisis para poder obtener un sistema estable cuando está en presencia de retardos y encontrar todas sus regiones de estabilidad.

6.3. REALIZAR UN ESTUDIO DE ESTABILIDAD UTILIZANDO CTCR PARA SISTEMAS CON RETARDOS

El capítulo principal de este trabajo consiste en el desarrollo del método de raíces características que incluya retardos que se simulan mediante software aplicando el procedimiento matemático que permite identificar las áreas de estabilidad del sistema utilizando la herramienta computacional matlab. Se realiza el análisis en dominio de tiempo de retardo por frecuencia cuando se tienen dos retardos y se aplica cada uno de los conceptos que posee este método, en donde se establece que poseen la misma frecuencia para luego pasar a un dominio de solo tiempo de retardos. Posteriormente se hace uso de una derivada parcial, que puede llegar a ser compleja dependiendo del tipo de sistema que se posea y así se determina la tendencia de esa raíz a estabilizar o desestabilizar el sistema con ese tiempo de retardo para poder determinar las zonas de estabilidad.

6.4. REALIZAR SIMULACIONES EN MATLAB DE 3 ROBOTS QUE SE COMUNIQUEN ENTRE SÍ LOS DATOS DE POSICIÓN Y VELOCIDAD

En esta etapa se realizó la respectiva simulación del sistema dinámico con su ley de control y sus componentes proporcionales y derivativos, esto con el fin de corroborar lo exacto y exhaustivo que es el método de agrupamiento de las raíces características, todo mediante el software MATLAB. Para ello fue necesario contar con una versión actualizada de este software el cual permitía tener una serie de funciones que se adaptaban a los cálculos requeridos; funciones como encontrar una raíz de una ecuación en un rango determinado. Para el código de simulación fue necesario crear una función para la matriz de Sylvester, cuyo objetivo es generar una función que permita graficar la *hipersuperficie constructora* y la *hipersuperficie de reflexión*. Después de ello tomar esa cantidad de puntos que conforman esas gráficas y usarlos en la ecuación característica para encontrar las frecuencias donde se genera una raíz sobre el eje imaginario y de esta manera encontrar los tiempos de retardo en el que esto ocurre.

Se utilizó la función de Matlab que resuelve ecuaciones diferenciales que permite incluir variables que representan los retardos en el tiempo. Esto fue de gran ayuda pues no fue necesario la implementación propia de este cálculo que hubiera aumentado la complejidad del software Los tiempos de retardo son escogidos por el usuario de acuerdo a la zona de estabilidad obtenida pudiendo así el usuario realizar simulaciones dentro o fuera de la zona de estabilidad según desee. Así se obtuvo un mapeo en 2D cuyos ejes corresponden al plano

cartesiano (x, y) y así observar y analizar cómo sería el comportamiento de estos robots al estar en presencia de retardos en sus comunicaciones.

6.5 CONSTRUCCIÓN DEL PROTOTIPO DE FORMACIÓN DE AGENTES EN PRESCENCIA DE RETARDOS

Se estudió la plataforma ICreate de robots no holonómicos para construirle la interfaz electrónica basada en un sistema embebido. En este sistema se programó la matemática de la formación de agentes en lenguaje c con Python simulando los retardos con delays. Se trazará una formación específica y los robots se monitorearán en tiempo real por comunicación xbee. Se comprobará la estabilidad del sistema bajo diferentes retardos para confrontarlo con las hipersuperficies de estabilidad obtenidas analíticamente del protocolo de agrupamiento de las raíces características.

6.6 INFORME RESULTADOS

En etapa final se recopilaron todos los resultados de las simulaciones y se verificó este método de agrupamiento de raíces características y su confiabilidad para el uso en robots cooperativos no holonómicos con tiempos de retardo en su comunicación. Por tanto en esta última etapa se resumieron los principales avances alcanzados en la investigación y se formularon conclusiones y recomendaciones de acuerdo con los resultados obtenidos.

En esta etapa se buscó plasmar en un informe final la documentación necesaria para que en trabajos futuros sea sencilla su aplicación o en dado caso su aplicación en otro tipo de proyectos. Se pretendió plantear además una metodología de aplicación del método de agrupamiento de raíces características para fácil comprensión del lector como también su implementación en el software de MATLAB, además de dar credibilidad al proyecto y su aplicación exacta y exhaustiva en sistemas dinámicos.

II. CAPITULO 2

7. PROTOCOLO DE CONSENSO EN PRESENCIA DE RETARDOS

Se analiza un método alternativo para el análisis de sistemas dinámicos con múltiples tiempos de retardos el cual se usará para analizar vehículos con restricciones no holonómicas bajo retardos en las comunicaciones. En este capítulo se realizará el debido procedimiento matemático para poder encontrar las zonas en el dominio del tiempo de retardo que generan estabilidad, donde se establecieron dos tiempos de retardo, uno en la variable de posición y otro en la variable de velocidad.

Para iniciar es necesario definir la topología de comunicación que será usada para que después de esta manera pueda ser definida la dinámica de cada agente que esta interactuando en la red.

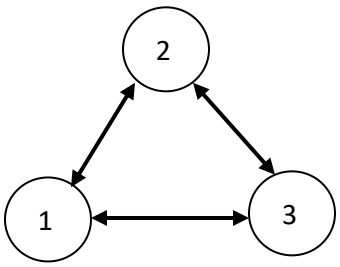


Figura 3. Topología de comunicación entre agentes.

PROTOCOLO DE CONSENSO PARA CADA AGENTE

La topología de comunicación escogida entre robots es descrita por un grafo que indica que la comunicación es bidireccional, así que cada agente es capaz de enviar y recibir información. De acuerdo a esto ahora es posible definir la dinámica de cada agente:

$$\ddot{x}_1 = P\left(\frac{x_2(t-\tau_1) + x_3(t-\tau_1)}{2} - x_1(t)\right) + D\left(\frac{\dot{x}_2(t-\tau_2) + \dot{x}_3(t-\tau_2)}{2} - \dot{x}_1(t)\right)$$

$$\ddot{x}_2 = P\left(\frac{x_1(t-\tau_1) + x_3(t-\tau_1)}{2} - x_2(t)\right) + D\left(\frac{\dot{x}_1(t-\tau_2) + \dot{x}_3(t-\tau_2)}{2} - \dot{x}_2(t)\right)$$

$$\ddot{x}_3 = P\left(\frac{x_1(t-\tau_1)+x_2(t-\tau_1)}{2} - x_3(t)\right) + D\left(\frac{\dot{x}_1(t-\tau_2)+\dot{x}_2(t-\tau_2)}{2} - \dot{x}_3(t)\right)$$

(7.1)

En (7.1) la dinámica de cada agente está basada en un protocolo de consenso en el que se promedian todo los valores de posición y velocidad de llegada de cada agente que se está comunicando con él y se resta su posición y velocidad con este promedio y se multiplica por una ganancia P proporcional para la posición y una ganancia D derivativa para la velocidad, con la particularidad de que los valores de posición tienen un retardo en el tiempo de valor constante igual a τ_1, y para la velocidad un valor constante igual a τ_2.

ESPACIO DE ESTADOS DEL SISTEMA DINAMICO

Para el análisis es claro que hay que obtener la ecuación característica de un sistema dinámico, para ello es necesario llevar el sistema a una representación de espacio de estados:

$$\dot{\mathbf{x}}(\mathrm{t}) = \left(\mathbf{I}_n \otimes \begin{bmatrix} 0 & 1 \\ -\mathrm{P} & -\mathrm{D} \end{bmatrix}\right)\mathbf{x}(t) + \left(\mathbf{C} \otimes \begin{bmatrix} 0 & 0 \\ \mathrm{P} & \mathrm{D} \end{bmatrix}\right)\mathbf{x}(t-\tau_1) + \left(\mathbf{C} \otimes \begin{bmatrix} 0 & 0 \\ 0 & \mathrm{D} \end{bmatrix}\right)\mathbf{x}(t-\tau_2)$$

(7.2)

Donde $\mathbf{x} = [x_1\dot{x}_1x_2\dot{x}_2 \cdots x_n\dot{x}_n]^T \in \mathrm{R}^{2n}$, $\mathbf{C}$ es la representación de la red de comunicación representada por $\mathbf{C} = \Delta^{-1}\mathbf{A}_\Gamma$, conformada por la multiplicación de la matriz de grado Δ, y la matriz de adyacencia $\mathbf{A}_\Gamma$, y ⊗ es el multiplicador de Kronecker [25].

TRANSFORMACIÓN LINEAL

Para un mejor análisis de este sistema dinámico es necesario realizar una transformación lineal planteada en [11], donde esta transformación permitirá analizar individualmente a cada agente que está interviniendo en la topología de comunicación. Para ello se introduce la siguiente transformación $\mathbf{x}(t) = (\mathbf{T} \otimes \mathbf{I_2})\boldsymbol{\xi}(t)$ en (7.2), donde $\mathbf{T} \in \mathrm{R}^{n\times n}$ es una matriz no singular. Con esto se obtiene:

$$\dot{\boldsymbol{\xi}}(t) = (\mathbf{T^{-1}} \otimes \mathbf{I_2})\left(\mathbf{I}_n \otimes \begin{bmatrix} 0 & 1 \\ -\mathrm{P} & -\mathrm{D} \end{bmatrix}\right)(\mathbf{T} \otimes \mathbf{I_2})\boldsymbol{\xi}(t) + (\mathbf{T^{-1}} \otimes \mathbf{I_2})\left(\mathbf{C} \otimes \begin{bmatrix} 0 & 0 \\ \mathrm{P} & \mathrm{D} \end{bmatrix}\right)(\mathbf{T} \otimes \mathbf{I_2})\boldsymbol{\xi}(t-\tau_1) +$$

$$(\mathbf{T^{-1}} \otimes \mathbf{I_2})\left(\mathbf{C} \otimes \begin{bmatrix} 0 & 0 \\ 0 & \mathrm{D} \end{bmatrix}\right)(\mathbf{T} \otimes \mathbf{I_2})\boldsymbol{\xi}(t-\tau_2)$$

(7.3)

Con la siguiente propiedad del multiplicador de Kronecker [25]:

$$(\mathbf{U} \otimes \mathbf{V})(\mathbf{W} \otimes \mathbf{Z}) = \mathbf{UW} \otimes \mathbf{VZ} \qquad (7.4)$$

La ecuación (7.3) llega a ser lo siguiente:

$$\dot{\boldsymbol{\xi}}(t) = \left(\mathbf{I}_n \otimes \begin{bmatrix} 0 & 1 \\ -\mathrm{P} & -\mathrm{D} \end{bmatrix}\right)\boldsymbol{\xi}(t) + \left(\boldsymbol{\Lambda} \otimes \begin{bmatrix} 0 & 0 \\ \mathrm{P} & \mathrm{D} \end{bmatrix}\right)\boldsymbol{\xi}(t-\tau_1) + \left(\boldsymbol{\Lambda} \otimes \begin{bmatrix} 0 & 0 \\ 0 & \mathrm{D} \end{bmatrix}\right)\boldsymbol{\xi}(t-\tau_2)$$

(7.5)

Donde $\boldsymbol{\Lambda}$ es una matriz conformada por los valores propios de $\mathbf{C}$ partiendo de que $\mathbf{T}^{-1}\mathbf{CT} = \boldsymbol{\Lambda}$. Así de esta manera podremos obtener n espacios de estados representando a cada agente:

$$\dot{\boldsymbol{\xi}}_j(t) = \begin{bmatrix} 0 & 1 \\ -\mathrm{P} & -\mathrm{D} \end{bmatrix}\boldsymbol{\xi}_j(t) + \lambda_j \begin{bmatrix} 0 & 0 \\ \mathrm{P} & \mathrm{D} \end{bmatrix}\boldsymbol{\xi}_j(t-\tau_1) + \lambda_j \begin{bmatrix} 0 & 0 \\ 0 & \mathrm{D} \end{bmatrix}\boldsymbol{\xi}_j(t-\tau_2)$$

(7.6)

DETERMINACIÓN ECUACIÓN CARACTERÍSTICA

Con $j = 1, 2, \ldots, n$. La ecuación característica completa del sistema podrá ser representada de la siguiente manera:

$$\det\left[s\mathbf{I} - \mathbf{A} - \sum_{l=1}^{p} \mathbf{B}_j \boldsymbol{e}^{-\tau_l s}\right] = \prod_{j=1}^{n} \det\left[s\mathbf{I_2} - \begin{bmatrix} 0 & 1 \\ -\mathrm{P} & -\mathrm{D} \end{bmatrix} - \lambda_j \begin{bmatrix} 0 & 0 \\ \mathrm{P} & \mathrm{D} \end{bmatrix}\boldsymbol{e}^{-\tau_1 s} - \lambda_j \begin{bmatrix} 0 & 0 \\ 0 & \mathrm{D} \end{bmatrix}\boldsymbol{e}^{-\tau_2 s}\right]$$

(7.7)

De acuerdo a lo anterior planteado lo único que queda por determinar es la matriz $\mathbf{C}$ correspondiente a la red de comunicación de la Figura [3] para poder determinar los valores propios del sistema y complementar la ecuación (7.7), ya que las ganancias P y D son escogidas por el usuario.

$$\mathbf{A_\Gamma} = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix}$$

$$\mathbf{C} = \Delta^{-1}\mathbf{A_\Gamma} = \begin{bmatrix} 0 & 1/2 & 1/2 \\ 1/2 & 0 & 1/2 \\ 1/2 & 1/2 & 0 \end{bmatrix}$$

$$\Delta = \begin{bmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 2 \end{bmatrix}$$

$$\mathbf{\Lambda} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1/2 & 0 \\ 0 & 0 & -1/2 \end{bmatrix}$$

(7.8)

Luego de haber obtenido todos los componentes de (7.7), es ahora posible determinar la ecuación característica del sistema como una multiplicación de términos correspondiendo cada término, a cada uno de los agentes de la red de comunicación.

Usando los valores de ganancia $\mathrm{P} = 1$ y $\mathrm{D} = 0.5$, aclarando que son escogidos aleatoriamente, siendo una ventaja del método CTCR ya que el resultado final se acomodara a estas ganancias y así se obtendrán las respectivas áreas de retardo donde el sistema es estable; la ecuación característica obtenida es la siguiente:

$$CE = \left(s^2 + 0.5s + 1 - 1(0.5se^{-\tau_2 s} + e^{-\tau_1 s})\right) \times$$
$$(s^2 + 0.5s + 1 + \frac{1}{2}(0.5se^{-\tau_2 s} + e^{-\tau_1 s})) \times (s^2 + 0.5s + 1 + \frac{1}{2}(0.5se^{-\tau_2 s} + e^{-\tau_1 s}))$$

(7.9)

8. CONTROL EN UNA FORMACION DE AGENTES

En la práctica los agentes que están participando en la topología de comunicación no pueden llegar al mismo punto, ya que esto causaría una colisión física entre ellos, por ello es necesario aplicar un control en formación, el cual se basa en mantener una distancia entre ellos al final del protocolo de consenso [21][22]. La forma para realizar un control en formación es agregar una constante o término forzado al sistema dinámico de cada agente; para este caso se analiza los subsistemas desacoplados planteados en (7.6). El procedimiento se inicia agregando a (7.6) el vector de términos forzados:

$$\dot{\boldsymbol{\xi}}_j(t) = \mathbf{A}_j\boldsymbol{\xi}_j(t) + \mathbf{B}_{1j}\boldsymbol{\xi}_j(t-\tau_1) + \mathbf{B}_{2j}\boldsymbol{\xi}_j(t-\tau_2) + \phi_j \qquad (8.1)$$

En donde $\mathbf{A}_j$, $\mathbf{B}_{1j}$ y $\mathbf{B}_{2j}$ se encuentran resumidos cada uno de los componentes que acompañaban a las variables de estado en (7.6), y $\phi_j = \begin{bmatrix}0 & \phi_j\end{bmatrix}^T$. De acuerdo a lo anteriormente planteado se procede a realizar la transformada de Laplace del subsistema para poder usar el teorema de valor final:

$$\boldsymbol{\xi}_j(s) = \left(s\mathbf{I} - \mathbf{A} - \mathbf{B}_{1j}e^{-s\tau_1} - \mathbf{B}_{2j}e^{-s\tau_2}\right)^{-1}\frac{\phi_j}{s} \qquad (8.2)$$

$$\boldsymbol{\xi}_{j,ss} = \lim_{s\to 0} s\boldsymbol{\xi}_j(s) = -\left(\mathbf{A} + \mathbf{B}_{1j} + \mathbf{B}_{1j}\right)^{-1}\phi_j$$

$$\boldsymbol{\xi}_{j,ss} = -\frac{\phi_j}{P(\lambda_j - 1)}\begin{bmatrix}1\\0\end{bmatrix} \qquad (8.3)$$

Usando la transformación lineal $\mathbf{x}_{ss} = (\mathbf{T}\otimes\mathbf{I_2})\boldsymbol{\xi}_{ss}$, donde $\mathbf{x}_{ss}$ es la distancia dada por el usuario al agente para que no colisione con algún otro agente. Despejando obtenemos $\boldsymbol{\xi}_{j,ss} = (\mathbf{T}^{-1}\otimes\mathbf{I_2})\mathbf{x}_{ss}$ y de esta manera encontrar los términos forzados ϕ_j. Posteriormente se usara $\mathbf{F} = (\mathbf{T}\otimes\mathbf{I_2})\boldsymbol{\Phi}$, donde $\mathbf{F} = [0\ f_1\ 0\ f_2\ 0\ f_3]^T$ y $\boldsymbol{\Phi} = [\phi_1^T\ \phi_2^T\ 0\ 0]^{\boldsymbol{T}}$. Hay que aclarar que el subsistema que tiene $\lambda = 1$, no puede tener un término forzado ya que este valor propio determina el centroide del sistema y si es forzado tendría un valor constante en la velocidad y llevaría todo el conjunto de agentes hacia el infinito. Por ello los dos primeros valores de $\boldsymbol{\Phi}$ son iguales a cero, que corresponden a este valor propio. La representación en espacio de estados quedaría de la siguiente manera:

$$\dot{\mathbf{x}}(\mathrm{t}) = \left(\mathbf{I}_3\otimes\begin{bmatrix}0 & 1\\ -\mathrm{P} & -\mathrm{D}\end{bmatrix}\right)\mathbf{x}(t) + \left(\mathbf{C}\otimes\begin{bmatrix}0 & 0\\ \mathrm{P} & \mathrm{D}\end{bmatrix}\right)\mathbf{x}(t-\tau_1) + \left(\mathbf{C}\otimes\begin{bmatrix}0 & 0\\ 0 & \mathrm{D}\end{bmatrix}\right)\mathbf{x}(t-\tau_2) + \mathbf{F} \qquad (8.4)$$

Para realizar el control en formación se usará la Figura 4 para el posicionamiento de los agentes, donde se escoge una cierta formación [22][23] la cual puede variar a condiciones dadas por el usuario y su aplicación, es decir la distancia entre cada agente puede ser diferente para su formación.

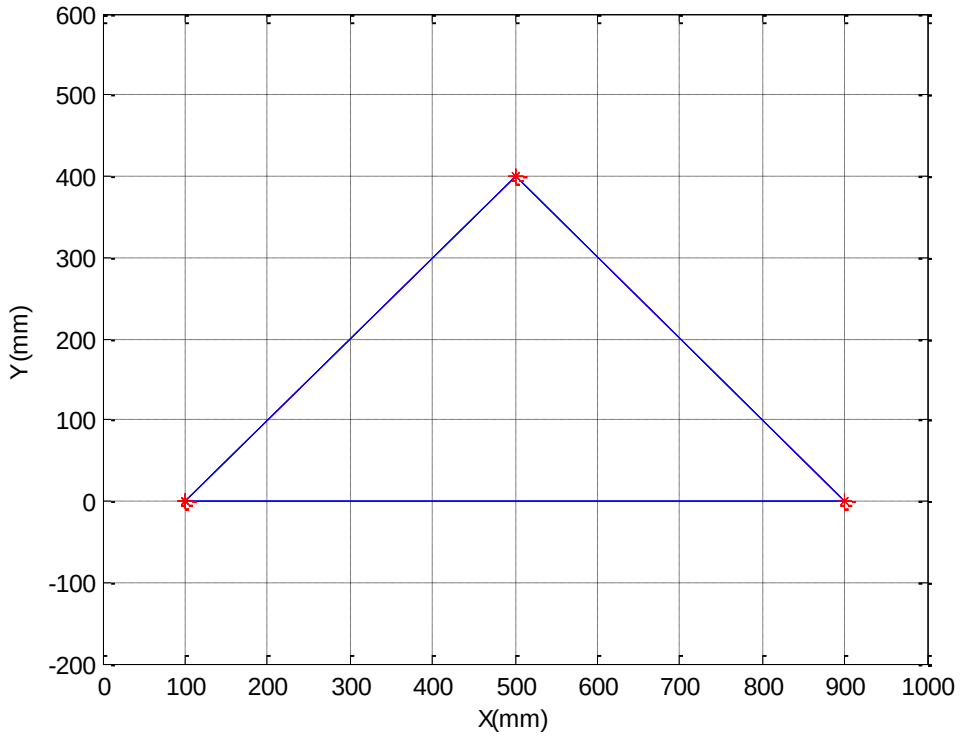


Figura 4.Ubicación de agentes.

De acuerdo a esto es posible definir las distancias finales que debe mantener cada agente en dirección x, $\mathbf{x}_{ss} = [0\ 100\ 0\ 500\ 0\ 900]^{\boldsymbol{T}}$ y en dirección y, $\mathbf{Y}_{ss} = [0\ 0\ 0\ 400\ 0\ 0]^{\boldsymbol{T}}$. De acuerdo a esto es posible obtener el valor final de los subsistemas de cada agente; en la dirección de x los términos son $\boldsymbol{\xi}_{ss}^{x} = [565.57\ 0\ 11.47\ 0\ 866.02\ 0]^{\boldsymbol{T}}$ y en la dirección de y $\boldsymbol{\xi}_{ss}^{y} = [6.62\ 0 - 326.53\ 0\ 230.94\ 0\ ]^{\boldsymbol{T}}$. Seguidamente se obtendrán los términos forzados mediante el uso de (8.3); obteniendo en la dirección de x, $\boldsymbol{\Phi}^{x} = [0\ 848.35\ 0\ 17.20\ 0\ 0]^{\boldsymbol{T}}$ y en dirección de y $\boldsymbol{\Phi}^{y} = [0\ 9.93\ 0 - 489.79\ 0\ 0\ 0]^{\boldsymbol{T}}$.
Haciendo uso de $\mathbf{F} = (\mathbf{T} \otimes \mathbf{I_2})\boldsymbol{\Phi}$ es posible encontrar el vector de términos forzados para el sistema dinámico obteniendo así $\mathbf{F}^{x} = [0 - 600\ 0\ 0\ 0\ 600]$ y $\mathbf{F}^{y} = [0 - 200\ 0\ 400\ 0\ -200]$. [10]

9. LINEALIZACIÓN POR REALIMENTACIÓN

En sistemas de control no lineal es frecuente encontrar problemas al momento de obtener los puntos estables del sistema, además que estos puntos pueden llevar el sistema a ser periódico. Por ello es comúnmente usado un método llamado control por realimentación el cual se basa en excitar el sistema con su mismo vector de estado con la particularidad de que puede ser multiplicado por una constante o poseer algún tipo de incertidumbre como un retardo en el tiempo; la idea básica es mantener el modelo matemático del sistema a partir de estas situaciones. [9]

La linealización por realimentación es una técnica de control que persigue linealizar y desacoplar parte de la dinámica de un sistema, permitiendo un control más sencillo, sin olvidar que se trata de una aproximación del verdadero sistema.

Una de las mejores aproximaciones la trabaja Oriolo et al [10] donde presenta una ley de control no lineal, variante con el tiempo, que hace que el sistema se vuelva equivalente a dos sistemas de segundo orden desacoplados. Estos sistemas de segundo orden consisten en integradores dobles, que pueden ser estabilizados por leyes de control lineales. Esto podrá ser comprendido a continuación.

Para la linealización por realimentación de (5.2), considere un nuevo vector de estado $\boldsymbol{Z} = [z_1, z_2]^T$. Este vector se selecciona de manera tal que las derivadas de las nuevas variables sean iguales a las de los estados x y y:

$$\begin{bmatrix} \dot{z_1} \\ \dot{z_2} \end{bmatrix} = \begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} = \begin{bmatrix} v\cos\theta \\ v\sin\theta \end{bmatrix} = v\begin{bmatrix} \cos\theta \\ \sin\theta \end{bmatrix} \qquad (9.1)$$

Tomando la segunda derivada del vector z se tiene:

$$\begin{bmatrix} \ddot{z_1} \\ \ddot{z_2} \end{bmatrix} = \begin{bmatrix} \dot{v}\cos\theta - v\omega\sin\theta \\ \dot{v}\sin\theta + v\omega\cos\theta \end{bmatrix} \qquad (9.2)$$

Siendo $\omega = \dot{\theta}$. Si la segunda derivada de z se hace igual a la entrada del sistema, se obtiene el equivalente linealizado:

$$\begin{bmatrix} \ddot{z}_1 \\ \ddot{z}_2 \end{bmatrix} = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \qquad (9.3)$$

Tras igualar (9.2) y (9.3), se obtiene una relación entre la entrada que se emplea para el sistema lineal y la del sistema no lineal:

$$\begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \begin{bmatrix} \cos\theta & -v\sin\theta \\ \sin\theta & v\cos\theta \end{bmatrix} \begin{bmatrix} \dot{v} \\ \omega \end{bmatrix} \qquad (9.4)$$

Invirtiendo la transformación (9.4) se puede calcular la entrada que debe ser aplicada al robot diferencial para que simule el comportamiento del sistema lineal [16]:

$$\begin{bmatrix} \dot{v} \\ \omega \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\frac{1}{v}\sin\theta & \frac{1}{v}\cos\theta \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \qquad (9.5)$$

Si se analiza (9.5) $u_1 = \ddot{x}$ y $u_2 = \ddot{y}$, siendo la relación del protocolo de consenso de cada robot con el sistema lineal para el control de un robot no holonómico, es decir, el protocolo de consenso que establece cada robot seria la entrada para controlar su velocidad lineal y angular del mismo. Esta entrada tiene incluido los tiempos de retardo y como ya se ha obtenido las zonas donde estos hacen estable un sistema en la Sección 7, (9.5) llegara a ser estable. Un aspecto importante de (9.5) es el hecho de que los términos de ω están limitados por v, ya que si este es igual a cero ω será indeterminado, esto es una característica de una restricción no holonómica, ya que estas están basadas en desigualdades y sus grados de libertad no son independientes. [18]

III. CAPITULO 3

10. EL ALGORITMO DE AGRUPAMIENTO DE RAICES CARACTERÍSTICAS (CTCR)

La construcción del método de CTCR puede ser presentado en dos proposiciones como es presentado en [12]. Para ello es necesario establecer tres definiciones:

Definicion1. Hipersuperficies Kernel φ_0: Son aquellos puntos del dominio del tiempo de retardo $\boldsymbol{\tau} \in R^{p+}$ para los que el sistema presenta una raíz imaginaria *s=wi* y que cumple con la siguiente restricción,
$0 < \tau_k < 2\pi/\omega$. Esto indica que este punto que pertenece al dominio del tiempo de retardo va acompañado de una frecuencia, que conjuntamente conforman una raíz en el eje imaginario del sistema dinámico. Estos puntos generan una gráfica que representa todas las raíces que se encuentran sobre el eje imaginario con el tiempo de retardo más pequeño posible.

Definicion2. Hipersuperficies descendientes φ: Las raíces puramente imaginarias de un sistema con retardos son periódicas con respecto al retardo. Por lo tanto si en cierto valor de tiempo de retardo se encuentra una raíz puramente imaginaria, es posible decir que si a ese tiempo de retardo le sumamos $2\pi/\omega$, donde ω es la frecuencia donde se genera esta raíz imaginaria, podría tener la misma raíz de este tipo en este nuevo tiempo de retardo:

$$< \tau_1 + \frac{2\pi}{\omega} j_1, \tau_2 + \frac{2\pi}{\omega} j_2, \dots, \tau_p + \frac{2\pi}{\omega} j_p > j_k = 1,2, \dots \qquad (10.1)$$

Definicion3. Tendencia de raíz, RT (Root Tendency): En algún punto del dominio del tiempo de retardo que pertenezca a φ_0 o a φ, donde se incremente infinitesimalmente algún tiempo de retardo, esto provocará que la raíz que se encuentre en el eje imaginario se desplace hacia la derecha o izquierda del eje imaginario del plano complejo. La tendencia de raíz RT indicará la dirección que tomará esta raíz, de acuerdo al incremento de alguno de los tiempos de retardo mientras los otros permanecen fijos.

$$RT|_{s=\omega i}^{\tau_j} = sgn\left[Re\left(\frac{\partial s}{\partial \tau_j}\right)\Big|_{s=\omega i}\right] \qquad (10.2)$$

Después de plantear las anteriores definiciones ahora es posible establecer las proposiciones para el CTCR.

Proposición 1. *Número finito de Hipersuperficies Kernel* para un dado sistema LTI-MTDS (*Linear Time Invariant Multiple Time Delay Systems*) presenta un número finito de hypersuperficies kernel, m, delimitado por el cuadrado del orden del sistema: $m < n^2$.

Proposición 2. *No varía la tendencia de raíz* para cada punto generado por $\boldsymbol{\tau} \in R^{p+}$, el cual ubica una raíz del sistema sobre el eje imaginario del plano complejo generando así las hipersuperficies kernel, de esta manera cuando se incrementa un tiempo de retardo dejando los restante tiempos de retardo fijos, se ubicará este sobre las hipersuperficies de descendencia y en ellas la tendencia de raíz permanecerá igual que en las hipersuperficies de kernel en esta dirección.

En este orden de ideas ahora es posible dar como aporte a este proyecto una serie de pasos en el que se pueda establecer un procedimiento para realizar el CTCR:

1. Se deben obtener las hipersuperficies kernel las cuales definen el tiempo más pequeño de retardo permitido por el sistema para obtener raíces sobre el eje imaginario. Posteriormente generar las hipersuperficies de descendencia usando la transformación no lineal planteada en (7.3).
2. Después de obtener las hipersuperficies kernel y descendencia debe ser usada la tendencia de raíz RT, esto con el fin de conocer hacia qué lado del plano complejo se dirigirá esta raíz si es incrementado infinitesimalmente un tiempo de retardo mientras los otros permanecen fijos.
3. Determinar la estabilidad del sistema cuando no esté bajo el dominio del tiempo de retardo, ya que este puede ser inestable antes de que se generen los tiempos de retardo. Este análisis es posible usando el criterio de Routh–Hürwitz.
4. Al obtener toda la información planteada por los puntos anteriores se debe hacer una tabla en la que se pueda establecer la cantidad de raíces inestables(NU) que se van acumulando cuando se va aumentando un tiempo de retardo mientras los otros permanecen fijos, ya que en dado punto puede que la RT sea negativa siendo esto favorable porque se podría creer que esta raíz tendería a estabilizarse, pero hay que tener en cuenta la raíces que se han venido acumulando, por ejemplo, en dado punto del dominio del tiempo de retardo se encuentran acumuladas una cantidad de raíces inestables(NU), pero en este punto se usa la RT y se obtiene que es negativa, quiere decir que hay cierta cantidad de raíces estables, estas se descontarán de la cantidad de raíces inestables, por lo tanto se puede concluir que en este punto todavía sigue siendo inestable a pesar que la RT dio negativa.
5. Declarar las regiones donde NU=0, siendo los tiempos de retardo de estas regiones los que permiten que el sistema sea estable.

Para sistemas con múltiples tiempos de retardos su respectivo análisis robusto de estabilidad es demasiado complejo, por ello también es posible usar el concepto del

SDS (Spectral Delay Space) [24] en el método de CTCR. Para su implementación es necesario realizar una construcción de bloques en el dominio del tiempo de retardo y bloques de longitud de 2π. Este mapeo por construcción de bloques convierte espectros sistemas de dimensión infinita en espectros de dimensión finita [13].

Este nuevo mapeo consiste en cambiar el dominio de $\boldsymbol{\tau} \in R^{p+}$ a un nuevo dominio $\boldsymbol{\vartheta} = \boldsymbol{\tau}\omega \in R^{p+}$, cambiando así la restricción que se mantenía para el mapeo por la siguiente $0 < \tau_k \omega < 2\pi$. Esto permite mantener todos los puntos generados por la hipersuperficie kernel dentro de un cubo de lado 2π que será llamado BB(Building Block), nombrando esto ahora como *hipersuperficie constructora* (δ_0^{SDS}). Como ya fue aclarado anteriormente existen unas hipersuperficies de descendencia debido a lo periódicas que se convierten las raíces de una ecuación característica situada sobre el eje imaginario. En el SDS estas hipersuperficies de descendencia serán llamadas como *hipersuperficies de reflexión* (δ^{SDS}), y serán bloques agrupados con una longitud de 2π. Una gran ventaja de este mapeo es el hecho de que es asegurado que solo se verán reflejados los puntos que generan una raíz sobre el eje imaginario dejando por fuera de la gráfica los que no generan estas raíces.

Para un mejor análisis de estabilidad y de cálculos para el empleo del método de CTCR, es necesario usar la transformación de Rekasius:

$$e^{-\tau_j s} = \frac{1-T_j s}{1+T_j s};\ T_j \epsilon Re \qquad (10.3)$$

Esta transformación de Rekasius hay que aclarar que no es similar a la aproximación de Padè, ya que es una transformación exacta. Si a esta transformación de Rekasius le realizamos un análisis de ángulo podremos encontrar lo siguiente:

$$\tau_j = \frac{2}{\omega}\left[\tan^{-1}\left(T_j\omega\right) + k\pi\right];$$

$$0 < \tan^{-1}\left(T_j\omega\right) \leq \pi, k = 0,1,2,\ldots \qquad (10.4)$$

$$T_j = \frac{\tan\left(\frac{\vartheta}{2}\right)}{\omega} = \frac{u_{\mathrm{j}}}{\omega};$$

$$\vartheta_j = \tau_j\omega \,\epsilon\, [0, 2\pi] \qquad (10.5)$$

Usando la ecuación (10.5) es posible transformar la ecuación característica (5.9) en una ecuación con términos más manejables y de esta manera evitar el uso de términos exponenciales. Se sustituye (10.5) en (7.9) para obtener lo siguiente:

$$P(\mathbf{u},\omega) = \overline{\mathrm{CE}}\left(s,\frac{\mathbf{u}}{\omega}\right)\Big|_{s=\omega i} = \sum_{k=0}^{N} b_k\left(\frac{\mathbf{u}}{\omega}\right)(\omega i)^k = 0 \qquad (10.6)$$

Donde $\mathbf{u} = (u_1, u_2, \ldots, u_p)$ y $P(\mathbf{u},\omega)$ es un polinomio en ω con coeficientes complejos que son parametrizados en **u.** Si allí existe una solución $\omega \in R$, ambas partes real e imaginaria deben ser zero simultanemante.

$$Re(P(\mathbf{u},\omega)) = \sum_{k=0}^{n} f_k(\mathbf{u})\,\omega^k = 0$$

$$Im(P(\mathbf{u},\omega)) = \sum_{l=0}^{n} g_l(\mathbf{u})\,\omega^l = 0 \qquad (10.7)$$

Es notorio que en las ecuaciones (10.4) y (10.5) hay un cambio de variable de *N* a *n*, esto es dado a los cambios de potencia que se presentan al momento de separar la parte real e imaginaria en dos funciones.

Es claro que las funciones real e imaginaria de (10.7) tienen un valor en común que es ω, por ello es posible relacionar estas funciones por medio de la matriz de Sylvester como es mostrado en (10.8) la cual llamaremos **M**.

$$\mathbf{M} = \begin{pmatrix} f_n(\mathbf{u}) & f_{n-1}(\mathbf{u}) & f_{n-2}(\mathbf{u}) & \cdots & \cdots & \cdots & \cdots & 0 \\ 0 & f_n(\mathbf{u}) & f_{n-1}(\mathbf{u}) & f_{n-2}(\mathbf{u}) & \cdots & \cdots & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \cdots & \cdots & \cdots & \cdots & f_2(\mathbf{u}) & f_1(\mathbf{u}) & f_0(\mathbf{u}) \\ g_n(\mathbf{u}) & g_{n-1}(\mathbf{u}) & g_{n-2}(\mathbf{u}) & g_{n-3}(\mathbf{u}) & \cdots & \cdots & \cdots & 0 \\ 0 & g_n(\mathbf{u}) & g_{n-1}(\mathbf{u}) & g_{n-2}(\mathbf{u}) & g_{n-3}(\mathbf{u}) & \cdots & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \cdots & \cdots & \cdots & g_3(\mathbf{u}) & g_2(\mathbf{u}) & g_1(\mathbf{u}) & g_0(\mathbf{u}) \end{pmatrix}_{2n\times 2n}$$

(10.8)

Si esta matriz de Sylvester se le es aplicado un determinante es posible obtener una función en términos de **u** como en (10.9), con la que podríamos graficar la *hipersuperficie constructora* dentro de un BB.

$$\det(\mathbf{M}) = F(\mathbf{u}) = F(\tan(\boldsymbol{\vartheta}/2)) = 0 \qquad (10.9)$$

A partir de estas nuevas ecuaciones planteadas y de un método para encontrar los puntos donde es posible encontrar raíces sobre el eje imaginario, se podrían plantear unos pasos para realizar la *hipersuperficie constructora* y la *hipersuperficie de reflexión*:

1. Por medio de algún programa para procedimientos matemáticos se debe generar la trayectoria del BB a partir de (10.9).

2. Con los puntos generados en la trayectoria encontrar $\omega \in R$, y los puntos que no cumplan con esto serán descartados.

3. Transformar desde *hipersuperficie constructora* en el dominio de $\boldsymbol{\vartheta}$, a la *hipersuperficie Kernel* en el dominio de $\boldsymbol{\tau}$, haciendo uso de la relación $\boldsymbol{\tau} = \boldsymbol{\vartheta}/\omega$ y así posteriormente seguir con el procedimiento del CTCR aplicando la RT y demás pasos que se plantearon anteriormente.

Partiendo de (7.9) de protocolo de consenso en presencia de retardos se va a aplicar el método anteriormente expuesto para la formación planteada en (8.4).

SUSTITUCIÓN DE REKASIUS

Seguidamente se procederá a aplicar el método de agrupamiento de las raíces características, aclarando que solo será aplicado al primer término de los tres que se están multiplicando en CE, por cuestiones de espacio.

Para el uso del método de CTCR se procede a introducir la ecuación (10.3) en (7.9), permitiendo usar la transformación de Rekasius y deja la ecuación característica sin términos exponenciales:

$$CE = s^2 + 0.5s + 1 - 1\left(0.5s\frac{1-T_2 s}{1+T_2 s} + \frac{1-T_1 s}{1+T_1 s}\right)=0$$

$$CE = s^2(1+T_2 s)(1+T_1 s) + 0.5s(1+T_2 s)(1+T_1 s) + 1(1+T_2 s)(1+T_1 s) - 0.5s(1-T_2 s)(1+T_1 s) - (1-T_1 s)(1+T_2 s)=0$$

(10.10)

Reemplazando $s = \omega i$ y usando (10.5) se pueden obtener las siguientes dos ecuaciones, una con términos reales y otra con términos imaginarios:

$$Re\big(CE(\mathbf{u},\omega)\big) = (u_2 u_1 - 1)\omega^2 - u_2\omega - 2u_2 u_1 = 0$$

$$Im\big(CE(\mathbf{u},\omega)\big) = -(u_1 + u_2)\omega^2 - u_2 u_1\omega + 2u_1 = 0$$

(10.11)

CONFORMACIÓN MATRIZ DE SYLVESTER

Una condición para que vayan a cero las ecuaciones de (10.11) es que tengan la misma frecuencia (ω), para ello una manera de relacionarlas y obtener una ecuación general para encontrar los valores de $\mathbf{u}$ es haciendo uso de la matriz de Sylvester (7.10):

$$\mathbf{M} = \begin{pmatrix} (u_2u_1 - 1) & -u_2 & -2u_2u_1 & 0 \\ 0 & (u_2u_1 - 1) & -u_2 & -2u_2u_1 \\ -(u_1 + u_2) & -u_2u_1 & 2u_1 & 0 \\ 0 & -(u_1 + u_2) & -u_2u_1 & 2u_1 \end{pmatrix} \qquad (10.12)$$

$$\begin{aligned} \det(\mathbf{M}) = F(\mathbf{u}) &= F\left(\tan\left(\frac{\boldsymbol{\vartheta}}{2}\right)\right) \\ &= -2\tan\left(\frac{\vartheta_1}{2}\right)^4 \tan\left(\frac{\vartheta_2}{2}\right)^4 - 2\tan\left(\frac{\vartheta_1}{2}\right)^3 \tan\left(\frac{\vartheta_2}{2}\right)^3 + 2\tan\left(\frac{\vartheta_1}{2}\right)^2 \tan\left(\frac{\vartheta_2}{2}\right)^4 \\ &\quad + 8\tan\left(\frac{\vartheta_1}{2}\right)^2 \tan\left(\frac{\vartheta_2}{2}\right)^2 + 4\tan\left(\frac{\vartheta_1}{2}\right)^2 - 2\tan\left(\frac{\vartheta_1}{2}\right)\tan\left(\frac{\vartheta_2}{2}\right)^3 = 0 \end{aligned}$$

(10.13)

HIPERSUPERFICIE CONSTRUCTORA E HIPERSUPERFICIE DE REFLEXIÓN

Con la ecuación (10.13) ahora es posible realizar una gráfica en términos de $\boldsymbol{\vartheta}$, gráfica que fue llamada como *hipersuperficie constructora* todo esto dentro de un BB. Para graficar esta ecuación se puede hacer uso de algún software computacional para cálculos matemáticos como Matlab.

A continuación se presenta la *construcción de hipersuperficie* de la CE planteada en (7.9):

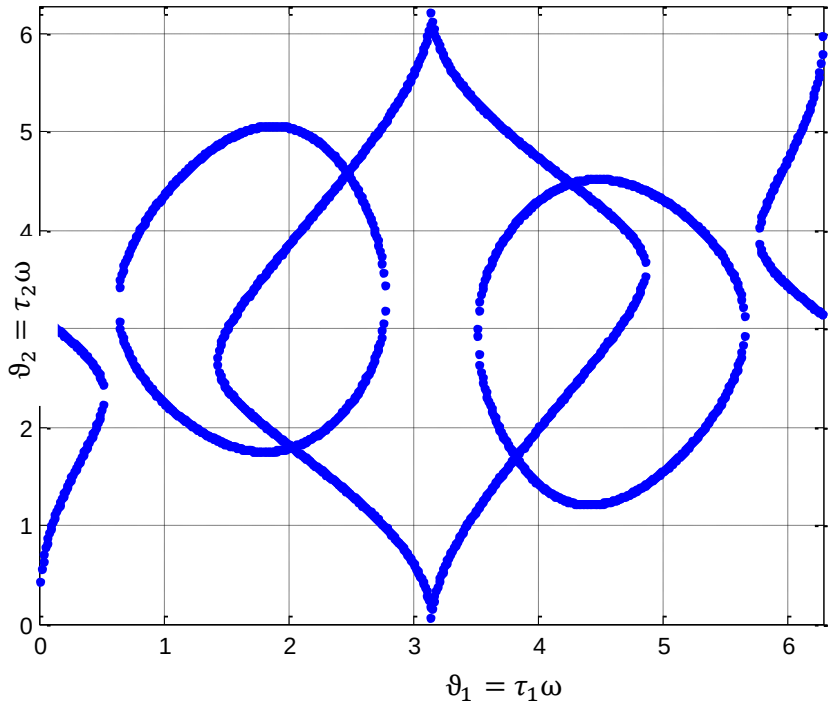


Figura 5. . Hipersuperficie constructora

Como se observa en la Figura 5, la *hipersuperficie constructora* se encuentra dentro de un cuadrado de bordes $2\pi \approx 6.28$ y está en términos de $\boldsymbol{\vartheta}$ viéndose reflejado el concepto de SDS. Los puntos ubicados en la gráfica generan una raíz sobre el eje imaginario del plano complejo.

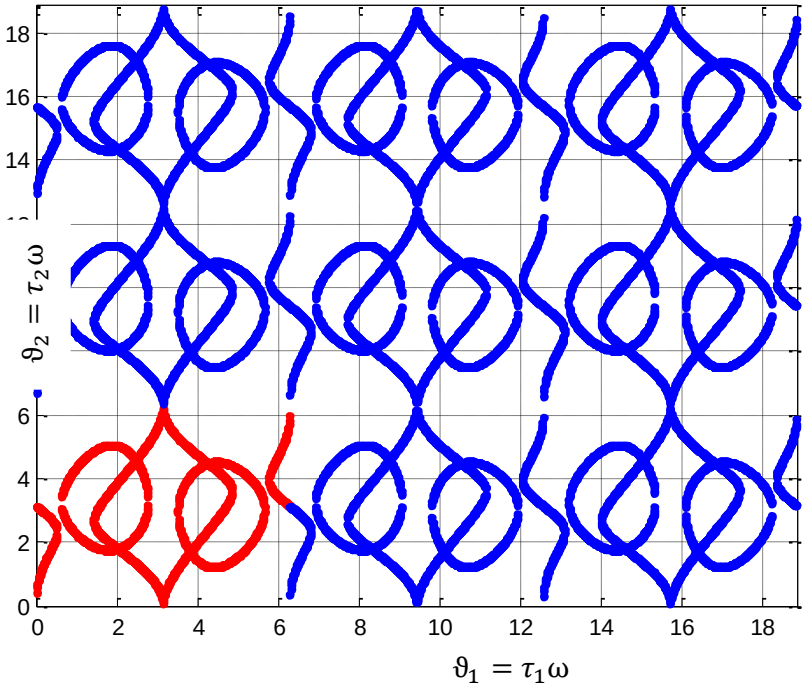


Figura 6. Hipersuperficies de reflexión

Uno de los aspectos más importantes a destacar es el uso del concepto de SDS es el hecho de demostrar que la gráfica es periódica como se evidencia en la Figura 6, en la que se puede observar en color rojo los puntos que conforman *hipersuperficie constructora* y en color azul la *hipersuperficies de reflexión*, siendo un punto a favor con su equivalencia a las *hipersuperficies de descendencia* del método de agrupamiento de las raíces características, hecho que ocurre debido a lo periódicas que son las raíces del sistema sobre el eje imaginario.

HIPERSUPERFICIE KERNEL Y DESCENDENCIA

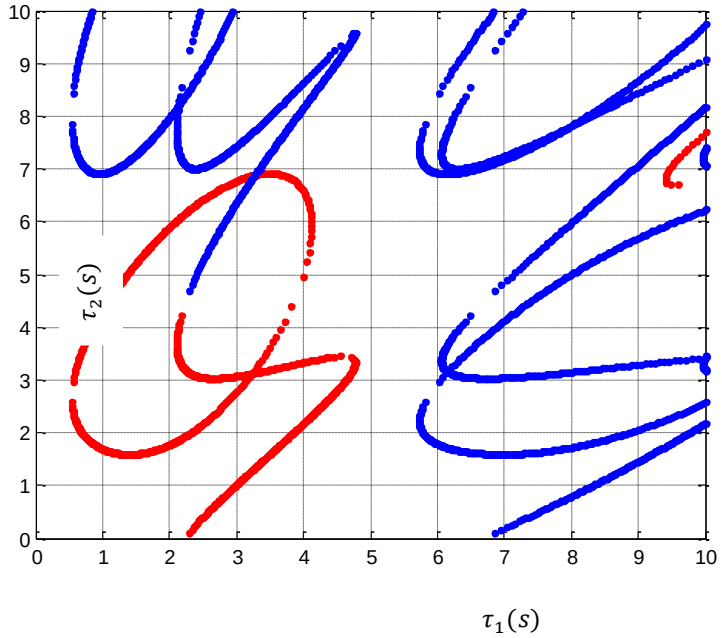


Figura 7. Hipersuperficie Kernel *e* hipersuperficies de Descendencia

Usando $\vartheta_j = \tau_j \omega$ es posible reaizar una transformación del dominio $\boldsymbol{\vartheta}$ al dominio del tiempo de retardo $\boldsymbol{\tau}$. En la Figura 7 se presentan los puntos en unidades de segundo en los que es generada una raíz del sistema sobre el eje imaginario. En color rojo se encuentran los puntos referentes a la *hipersuperficie kernel*, y en color azul los que representa la *hipersuperficies de descendencia*.

Ya obtenida la gráfica en el dominio del tiempo de retardo, como posterior análisis es necesario aplicar RT con el fin de conocer hacia que parte del plano complejo se dirige, es decir, si hacia el semiplano derecho convirtiéndose en una raíz inestable o hacia el semiplano izquierdo convirtiéndose en una raíz estable.

TENDENCIA DE RAÍZ

Como primer análisis se aplicará RT a la ecuación característica variando respecto al tiempo de retardo τ_1. Por cuestiones prácticas solo será aplicado al primer término de CE en (7.9):

$$\frac{\partial s}{\partial \tau_1} = \frac{-se^{-s\tau_1}}{2s + 0.5 - 0.5e^{-s\tau_2} + 0.5\tau_2 se^{-s\tau_2} + \tau_1 e^{-s\tau_1}}$$

$$RT|_{s=\omega i}^{\tau_1} = sgn\left[Re\left(\frac{\partial s}{\partial \tau_1}\right)\Big|_{s=\omega i}\right] \qquad (10.14)$$

Después de obtener (7.14) se procede a aplicar esta ecuación a cada uno de los puntos obtenidos en la Figura 7, generando la siguiente gráfica:

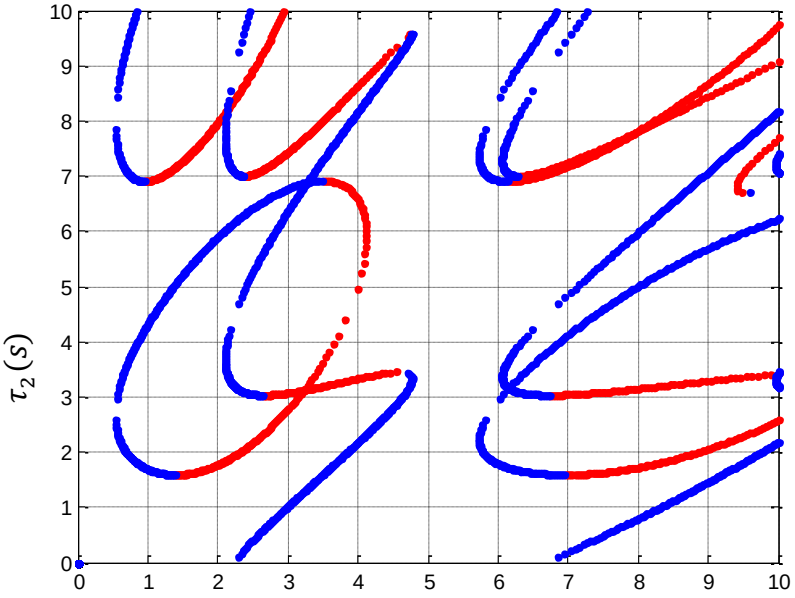


Figura 8. *Uso de RT con respecto a τ_1*

En la Figura 8 se encuentran en color rojo los puntos que corresponden al tiempo de retardo que generaran una raíz $\tau_1(s)$ estable al aumentar infinitesimalmente τ_1, y en azul los que generan una raiz inestable.

Ahora se aplicará RT a la ecuación característica variando respecto al tiempo de retardo τ_2:

$$\frac{\partial s}{\partial \tau_2} = \frac{-0.5s^2 e^{-s\tau_2}}{2s + 0.5 - 0.5e^{-s\tau_2} + 0.5\tau_2 s e^{-s\tau_2} + \tau_1 e^{-s\tau_1}}$$

$$RT|_{s=\omega i}^{\tau_2} = sgn\left[Re\left(\frac{\partial s}{\partial \tau_2}\right)\Big|_{s=\omega i}\right] \qquad (10.15)$$

Nuevamente se aplicará (10.15) a los puntos de la Figura 8, generando lo siguiente:

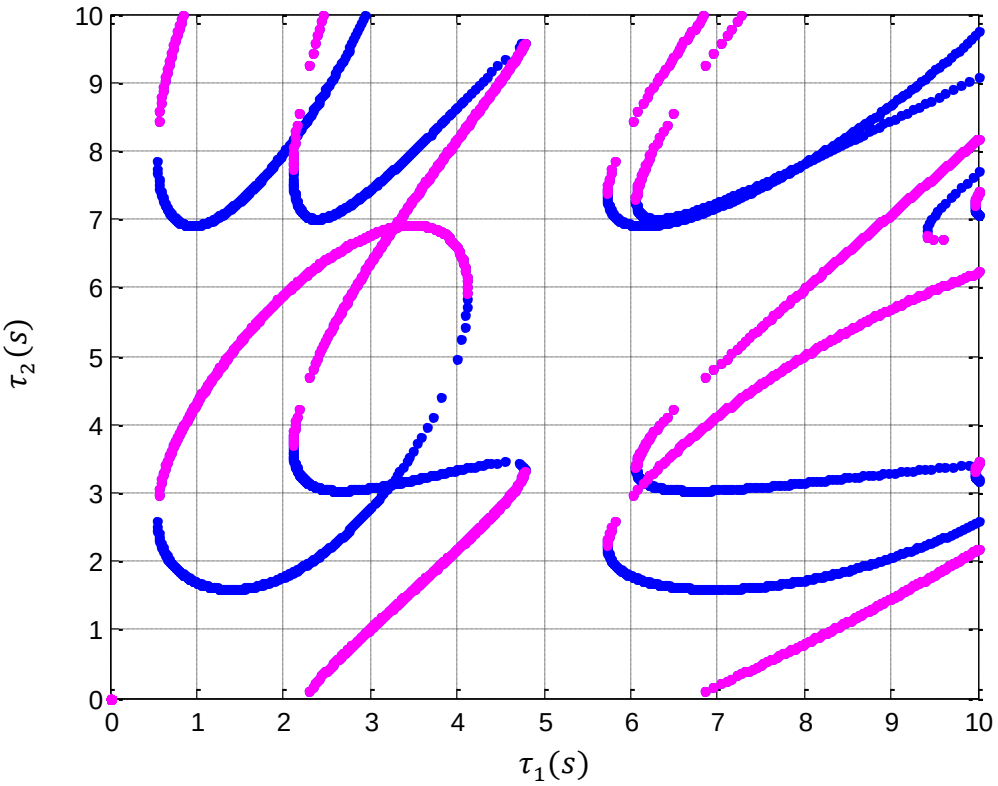


Figura 9. Uso de RT con respecto a τ_2

En la Figura 9 se encuentran en color magenta los puntos que corresponden al tiempo de retardo que generaran una raíz estable al aumentar infinitesimalmente τ_2, y en azul los que generan una raíz inestable.

Tomando la Figura 8 y la Figura 9 ahora es posible definir la zonas con tiempo de retardo que generan estabilidad para el sistema. La gráfica de las zonas estables es la siguiente:

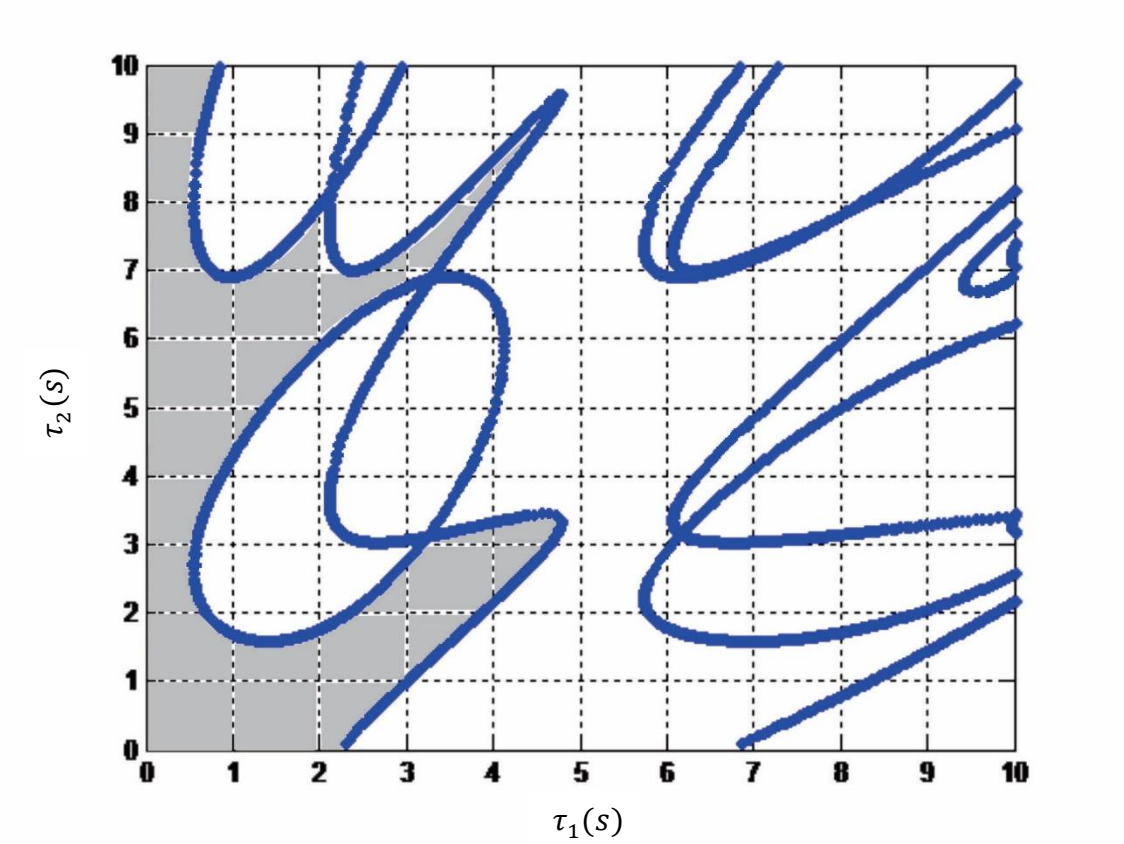


Figura 10. Regiones estables para el sistema dado.

En la Figura 10 se puede observar sombreadas las zonas que generan estabilidad al sistema dinámico en presencia de retardos.

11. SIMULACIÓN COMPUTACIONAL DE ESTABILIDAD EN CONCENSO DE ROBOTS NO HOLONÓMICOS

El proceso de simulación fue realizado mediante el software MATLAB, el cual es explicado en diagrama de bloques en la figura 11

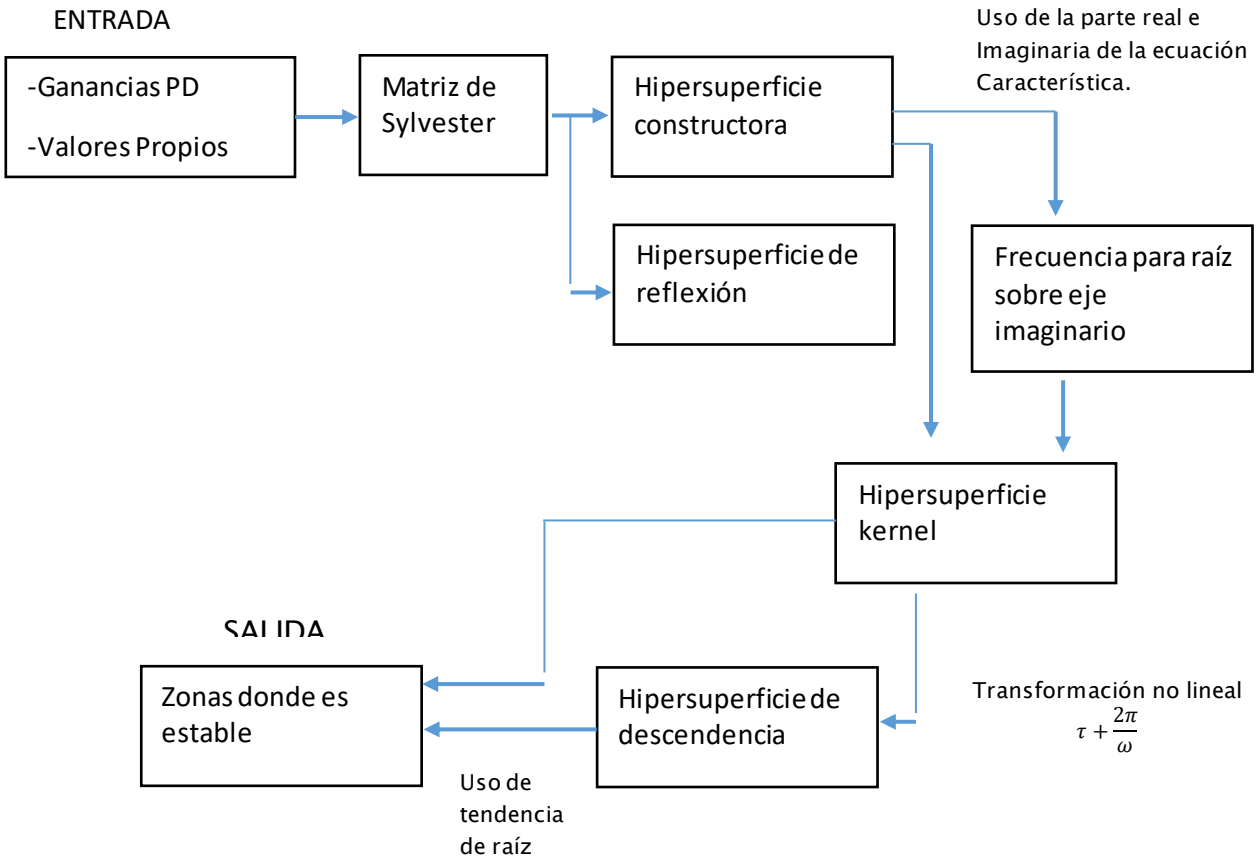


Figura 11.Diagrama de bloques.

La explicación de los diagramas de bloques se encuentra a continuación:

1. El primer paso es obtener como valores de entrada las ganancias *PD* y los valores propios del análisis realizado al sistema dinámico (7.1). Para la topología de la Figura 3 se obtuvieron los valores propios 1, -0.5, -0.5, y se escogieron unas ganancias *P=1 y D=0.5.*

2. El primer proceso se realiza con el primer valor propio ingresado. Luego se debe obtener el análisis del sistema dinámico en términos de $0 < \boldsymbol{\vartheta} < 2\pi$. El sistema dinámico tratado posee dos tiempos de retardo por lo tanto posee ϑ_1

y ϑ_2, así que en el código se inicia dando un valor de ϑ_1 y con ese se obtiene el valor de ϑ_2. Para este caso ϑ_1 inicia con un valor de 0.1 e incrementa en un paso de 0.1 hasta $n\pi$. El término ϑ_2 es obtenido haciendo uso de la matriz de Sylvester.

```
P=1; %ganancia P
D=0.5; %ganancia D
vap=[1 -0.5]; %valor propio a evaluar
k=0;
kk=0;
f=0;
a=0;
d=0;
rad1=6;%cantidad de radianes para encontrar puntos
g=length(vap);

%=========================================================
%  Busqueda de valores v1 y v2
%=========================================================
for z=1:g
    vp=vap(z);

for rad=2:2:rad1
for v1=0.01:0.01:(n*pi) %valores de v1 hasta n*pi

      u1=tan(v1/2);
      q=Sylvester(u1,P,D,vp);
```

Figura 12.Condiciones de entrada.

3. Esta matriz de Sylvester es realizada en una función de MATLAB la cual es invocada. Esta función posee los valores reales e imaginarios establecidos en (7.12); a esta matriz se le es aplicado un determinante del cual resulta una ecuación, la cual es usada para obtener ϑ_2.

```
function x= Sylvester(u1,P,D,vp)
% function x= Sylvester(u1)
syms u2
 S=[(u2*u1-1) D*(u1*(vp-1)-u2*(vp+1)) P*(-u2*u1*(vp+1)-vp+1) 0;... %matriz de silvester
     0 (u2*u1-1) D*(u1*(vp-1)-u2*(vp+1)) P*(-u2*u1*(vp+1)-vp+1);...
     -(u1+u2) D*(-u2*u1*(vp+1)-vp+1) P*(u1*(1+vp)+u2*(1-vp)) 0;...
     0 -(u1+u2) D*(-u2*u1*(vp+1)-vp+1) P*(u1*(1+vp)+u2*(1-vp))];

      d=(det(S)); %determinante de la matriz de silvester, este
                  %determinante esta en terminos de u2
                  %ya que u1 fue previamente dado. aca se obtiene un
                  %polinomio cuyas variable es u2
       x=vpasolve(d,u2);%encuentra los valores de las raices para u2,
                        %pero no son todavia los valores reales de v2.
                       %aca se encuentran las raices de un polinomio

```

Figura 13.Uso de función de matriz de Sylvester.

4. Hay que tener en cuenta que hay un concepto inicial que es la *hipersuperficie constructora* el cual se basa en encontrar los valores $0 < \boldsymbol{\vartheta} < 2\pi$, para ello se usa la función *vpasolve* la cual se encargará de encontrar estos valores en un rango dado y así se irán guardando en una matriz. Los valores que estén fuera de este rango pertenecen a la *hipersuperficie de reflexión* cuyos valores irán guardándose en otra matriz.

```
35              l=vpasolve(u==q(i),v2,[(rad-2)*pi rad*pi]); %la raiz real que se
36                                                  %encontro de u2 se iguala a
37                                                  %"u" para ahora si encontrar
38                                                  %los valoresde v2 en un
39                                                  %rango de 0 a 2*pi
40                  if(v1<=(2*pi))&&(l<=(2*pi))
41                      k=k+1; %usado para ir colocando los valores encontrados
42                             %en un vector llamado
43                             %"construccion_hypersuperficies"
44                      construccion_hypersuperficies(k,1)=v1; %se usa para
45                       %imprimir los valores de v1 y v2 con el fin de
46                       %analizarlos si es necesario
47
48                      construccion_hypersuperficies(k,2)=l;
49                      construccion_hypersuperficies(k,3)=vp;
50                      construccion_hypersuperficies_v1(k)=v1; %vectores de v1
51                                                              %y de v2 para poder
52                                                              %graficarlos
53                      construccion_hypersuperficies_v2(k)=l;
54                      construccion_hypersuperficies_vp(k)=vp;
55
56                  else
57                      kk=kk+1; %usado para ir colocando los valores encontrados
58                               %en un vector llamado
59                               %"reflexion_hypersuperficies"
60                      reflexion_hypersuperficies(kk,1)=v1; %se usa para
61                                    %imprimir los valores de v1 y v2
62                                    %con el fin de analizarlos si es necesario
63                      reflexion_hypersuperficies(kk,2)=l;
64                      reflexion_hypersuperficies(kk,3)=vp;
65
66                      reflexion_hypersuperficies_v1(kk)=v1; %vectores de v1 y
67                                                          %de v2 para poder graficarlos
68                      reflexion_hypersuperficies_v2(kk)=l;
69                      reflexion_hypersuperficies_vp(kk)=vp;
70                  end
```

Figura 14.Hipersuperficie constructora e Hipersuperficie de reflexión.

5. Seguidamente es necesario ahora obtener las frecuencias donde se generan raíces sobre el eje imaginario. Para ello se usa la matriz creada con los datos de *hipersuperficie constructora* y haciendo uso de las ecuaciones (7.11) las cuales tienen en común ω. Se resuelven las ecuaciones de (7.11) encontrando sus raíces, para luego comparar las raíces encontradas en cada ecuación. La frecuencia real que sea igual en ambos resultados será la frecuencia donde se generará una raíz sobre el eje imaginario. Nuevamente serán guardados estos datos en una matriz con la diferencia que se anexara su frecuencia.

```
%=========================================================================
%  Busqueda de frecuencia(w) para una raiz imaginaria
%=========================================================================
   for i=1:k

   v1=construccion_hypersuperficies_v1(i);
   v2=construccion_hypersuperficies_v2(i);
   vp=construccion_hypersuperficies_vp(i);
   u1=tan(v1/2);
   u2=tan(v2/2);

   w1=(u2*u1-1)*w^2 + D*(u1*(vp-1)-u2*(vp+1))*w + P*(-u2*u1*(vp+1)-vp+1);
%    w1=(u1*u2-1)*w^2+ (u1-7*u2)*w+ (-4*u1*u2+22);
   w1raices=vpasolve(w1,w);

   w2=-(u1+u2)*w^2 + D*(-u2*u1*(vp+1)-vp+1)*w + P*(u1*(1+vp)+u2*(1-vp));
% w2=-(u1+u2)*w^2+ (5-u1*u2)*w+ (10*u1-4*u2);
   w2raices=vpasolve(w2,w);
   f=f+1;
   vector(f,1)=w1raices(1);
   vector(f,2)=w2raices(1);
   vector(f,3)=v1;
   vector(f,4)=v2;
   vector(f,5)=vp;
   f=f+1;
   vector(f,1)=w1raices(2);
   vector(f,2)=w2raices(2);
   vector(f,3)=v1;
   vector(f,4)=v2;
   vector(f,5)=vp;
   end

 datos=double(vector);
```

Figura 15.Busqueda de frecuencia para raíz sobre el eje imaginario.

6. El siguiente paso es encontrar el tiempo de retardo mínimo o también llamada la *hipersuperficie kernel* donde se encuentran las raíces sobre el eje imaginario. Es posible ya que en este punto se cuenta con la frecuencia en la que ocurre y así complementar la siguiente ecuación $\tau_j = \frac{\vartheta_j}{\omega}$.

```
%=========================================================================
%  Busqueda de tiempos de retardo para kernel
%=========================================================================
 for l=1:2:f
        if (datos(l,1)==datos(l,2))
            a=a+1;

            datos_kernel(a,1)=datos(l,3)/datos(l,1);%tiempo de retardo t1
            datos_kernel(a,2)=datos(l,4)/datos(l,1);%tiempo de retardo t2
            datos_kernel(a,3)=datos(l,1);%frecuencia de retardo
            datos_kernel(a,4)=datos(l,5);%valor de vp
        end

        if (datos(l,1)==datos(l+1,2))
            a=a+1;

            datos_kernel(a,1)=datos(l,3)/datos(l,1);%tiempo de retardo t1
            datos_kernel(a,2)=datos(l,4)/datos(l,1);%tiempo de retardo t2
            datos_kernel(a,3)=datos(l,1);%frecuencia de retardo
            datos_kernel(a,4)=datos(l,5);%valor de vp
        end

        if (datos(l+1,1)==datos(l,2))
            a=a+1;

            datos_kernel(a,1)=datos(l,3)/datos(l+1,1);%tiempo de retardo t1
            datos_kernel(a,2)=datos(l,4)/datos(l+1,1);%tiempo de retardo t2
            datos_kernel(a,3)=datos(l+1,1);%frecuencia de retardo
            datos_kernel(a,4)=datos(l,5);%valor de vp
        end

        if (datos(l+1,1)==datos(l+1,2))
            a=a+1;
             datos_kernel(a,1)=datos(l,3)/datos(l+1,1);%tiempo de retardo t1
            datos_kernel(a,2)=datos(l,4)/datos(l+1,1);%tiempo de retardo t2
            datos_kernel(a,3)=datos(l+1,1);%frecuencia de retardo
            datos_kernel(a,4)=datos(l,5);%valor de vp
        end
 end
```

Figura 16.Hipersuperficie kernel

7. Como fue nombrado en secciones anteriores los tiempos de retardo que conforman la hipersuperficie kernel son periódicos y se repiten cada $\frac{2\pi}{\omega}$, por lo que es posible usar los datos de la matriz hipersuperficie kernel y realizarle esta transformación no lineal para obtener las hipersuperficies de descendencia.

```
%=========================================================================
%  Busqueda de tiempos de retardo para descendencia de kernel
%=========================================================================

for incremento=2:2:(rad1-2)
 for i=1:1:a
    d=d+1;
    descendencia_kernel(d,1)=datos_kernel(i,1)+((incremento*pi)/datos_kernel(i,3));
    descendencia_kernel(d,2)=datos_kernel(i,2);
    descendencia_kernel(d,3)=datos_kernel(i,3);
    descendencia_kernel(d,4)=datos_kernel(i,4);
 end
end

for incremento=2:2:(rad1-2)
  for i=1:1:a
    d=d+1;
    descendencia_kernel(d,1)=datos_kernel(i,1);
    descendencia_kernel(d,2)=datos_kernel(i,2)+((incremento*pi)/datos_kernel(i,3));
    descendencia_kernel(d,3)=datos_kernel(i,3);
    descendencia_kernel(d,4)=datos_kernel(i,4);

     bandera=d;
    for incremento2=2:2:(rad1-2)
      d=d+1;
      descendencia_kernel(d,1)=descendencia_kernel(bandera,1)+((incremento2*pi)/descendencia_kernel(bandera,3));
      descendencia_kernel(d,2)=descendencia_kernel(bandera,2);
      descendencia_kernel(d,3)=descendencia_kernel(bandera,3);
      descendencia_kernel(d,4)=descendencia_kernel(bandera,4);
    end
  end

end
```

Figura 17. Hipersuperficie de descendencia.

8. Después de obtener la hipersuperficie kernel y las hipersuperficies de descendencia seguidamente se debe realizar el análisis de tendencia de raíz RT, esto con el fin de encontrar la zona de estabilidad. Para ello se realiza la derivada parcial de le ecuación característica con respecto a un tiempo de retardo, esto se realizó a mano y fue posteriormente agregado a al código para poder evaluar. Se usa la función real la cual se encarga de tomar solo la parte real de esta evaluación para luego verificar que si esta parte real es mayor que cero, esta raíz tenderá a desestabilizar el sistema y si es menor que cero esta tenderá a estabilizar el sistema dinámico.

```
%=========================================================================
%  Tendencia de raiz para kernel
%=========================================================================
  for r=1:1:a
            t1=datos_kernel(r,1);
            t2=datos_kernel(r,2);
            w=datos_kernel(r,3);
            vp=datos_kernel(r,4);

            numerador1=-vp*P*(w*1i)*exp(-1i*t1*w);
            denominador1=2*(w*1i)+D-vp*D*exp(-t2*(w*1i))+vp*D*t2*(w*1i)*exp(-t2*(w*1i))+vp*P*t1*exp(-t1*(w*1i));

            derivada1=numerador1/denominador1;
            RT1=real(derivada1);
            numerador2=-vp*D*(w*1i)^2*exp(-t2*(w*1i));
            denominador2=2*(w*1i)+D-vp*D*exp(-t2*(w*1i))+vp*D*t2*(w*1i)*exp(-t2*(w*1i))+vp*P*t1*exp(-t1*(w*1i));
            derivada2=numerador2/denominador2;
            RT2=real(derivada2);
            if (RT1>0)
                inestables_kernel_t1_RT1(r)=t1;
                inestables_kernel_t2_RT1(r)=t2;
            end
            if (RT1<0)
                estables_kernel_t1_RT1(r)=t1;
                estables_kernel_t2_RT1(r)=t2;
            end
            if (RT2<0)
                estables_kernel_t1_RT2(r)=t1;
                estables_kernel_t2_RT2(r)=t2;
            end
  end
```

Figura 18. Tendencia de raíz.

9. Como parte final del código se obtienen las gráficas necesarias para el análisis y representación del método de CTCR. En la Figura19 se presenta el código para graficar la *construcción de hipersuperficie*, *la reflexión de hipersuperficies* y la *hipersuperficie kernel*.

```
figure(1)
  plot(construccion_hypersuperficies_v1,construccion_hypersuperficies_v2,'.b')
  hold on
  grid on
  axis([0,2*pi,0,2*pi])

figure(2)
  plot(construccion_hypersuperficies_v1,construccion_hypersuperficies_v2,'.r')
  hold on
  grid on
  axis([0,rad1*pi,0,rad1*pi])
figure(2)
  plot(reflexion_hypersuperficies_v1,reflexion_hypersuperficies_v2,'.b')
  hold on
  grid on

 figure(3)
  plot(datos_kernel(1:end,1),datos_kernel(1:end,2),'.r')
  hold on
  grid on
  axis([0,10,0,10])
  plot(descendencia_kernel(1:end,1),descendencia_kernel(1:end,2),'.b')
```

Figura 19. Graficas de Hipersuperficies.

En la Figura20 se presenta el código para graficar las tendencias de raíz que permiten que el sistema sea estable, también se grafica las hipersuperficies de descendencia y las tendencias de raíz inestables.

```
figure(4)
 plot(estables_kernel_t1_RT1,estables_kernel_t2_RT1,'.r')
 hold on
 grid on
 axis([0,10,0,10])
 hold on
 plot(estables_descendencia_kernel_t1_RT1,estables_descendencia_kernel_t2_RT1,'.r')
 hold on
 plot(inestables_kernel_t1_RT1,inestables_kernel_t2_RT1,'.b')
 hold on
 plot(inestables_descendencia_kernel_t1_RT1,inestables_descendencia_kernel_t2_RT1,'.b')


 figure(5)
 plot(datos_kernel(1:end,1),datos_kernel(1:end,2),'.b')
 hold on
 grid on
 axis([0,10,0,10])
 plot(descendencia_kernel(1:end,1),descendencia_kernel(1:end,2),'.b')
 hold on
 plot(estables_kernel_t1_RT2,estables_kernel_t2_RT2,'.m')
 hold on
 plot(estables_descendencia_kernel_t1_RT2,estables_descendencia_kernel_t2_RT2,'.m')

 figure(6)
 plot(datos_kernel(1:end,1),datos_kernel(1:end,2),'.b')
 hold on
 grid on
 axis([0,10,0,10])
 plot(descendencia_kernel(1:end,1),descendencia_kernel(1:end,2),'.b')
```

Figura 20. *Graficas de tendencia de raiz.*

IV. CAPÍTULO 4

12. ANÁLISIS DE RESULTADOS

Con las zonas de estabilidad en el dominio de tiempo de retardo obtenido en la Figura6, ahora es posible simular el sistema con retardos mediante un software matemático como lo es Matlab; todo esto con el fin de corroborar el procedimiento y análisis realizado. Para ello se escogerá de la Figura 9 los valores de tiempo de retardo $(\tau_1, \tau_2) = (0.5s, 0.5s)$, valores que se encuentran en una zona que representa estabilidad en el protocolo de consenso:

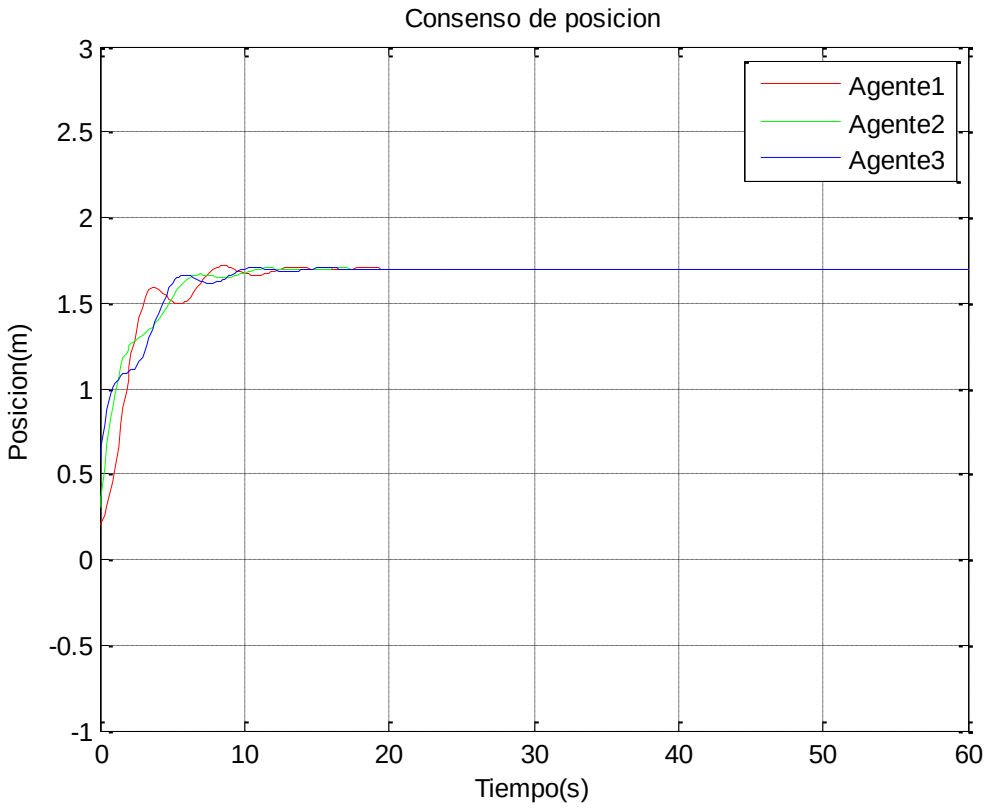


Figura 21. Consenso de posición unidimensional estable.

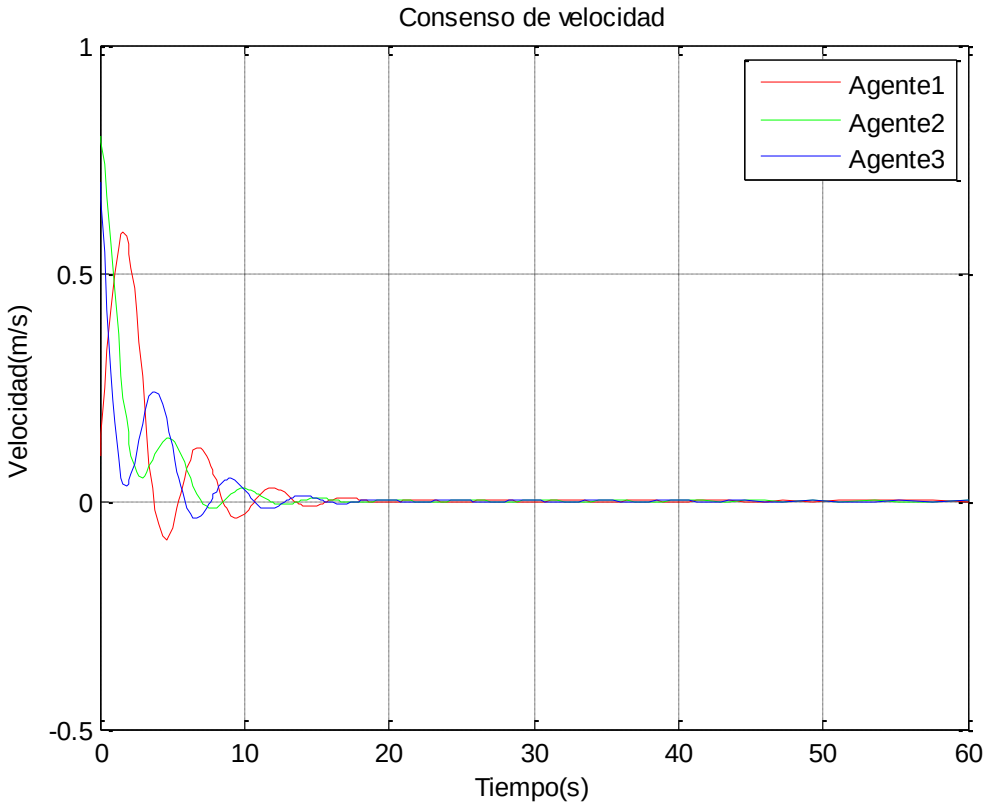


Figura 22. Consenso de magnitud de velocidad estable.

Como se evidencia en la Figura 22 los agentes llegan a un punto en común en posición y en cuanto a su velocidad llegan a un valor igual a cero como se observa en la Figura 13. Ahora se escogerá de la Figura 9 el punto $(\tau_1, \tau_2) = (1s, 3s)$, punto que según el análisis realizado genera inestabilidad para el protocolo de consenso:

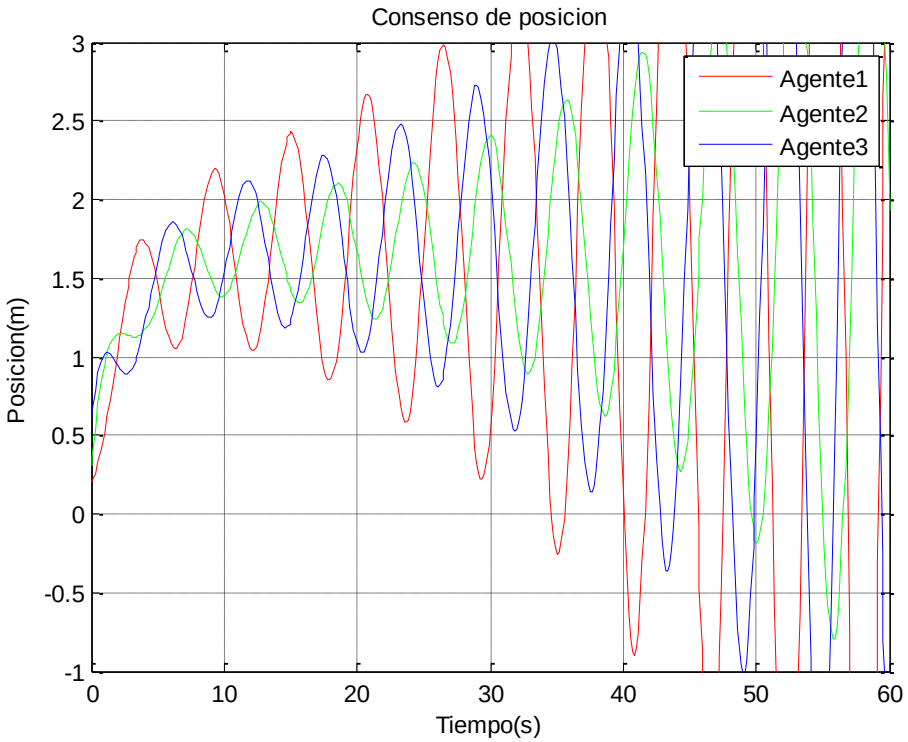


Figura 23. Consenso de posición unidimensional inestable.

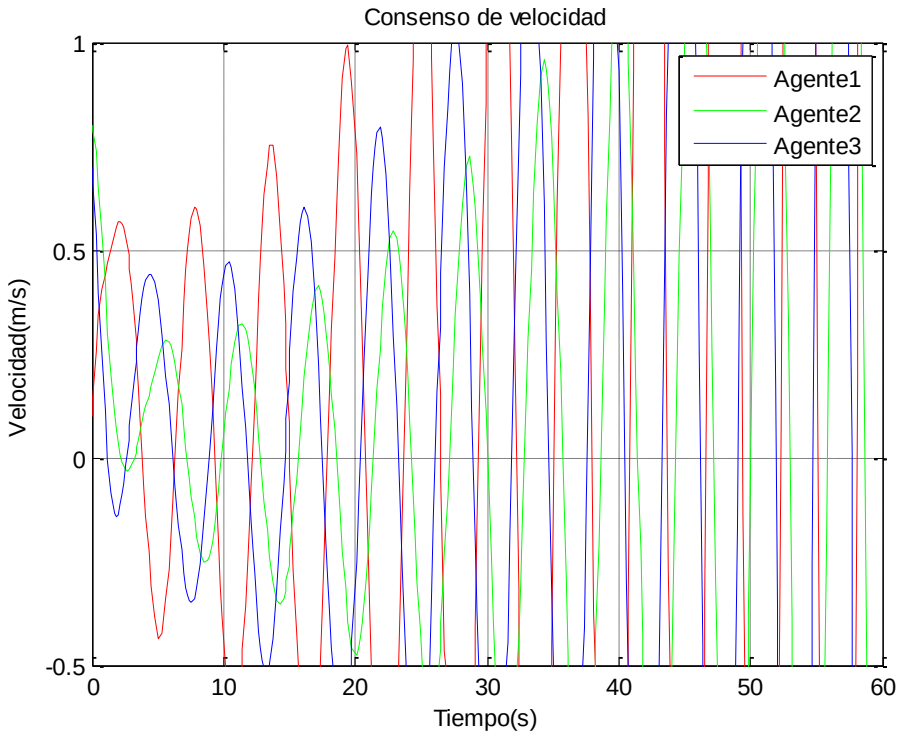


Figura 24. Consenso de magnitud de velocidad inestable.

La Figura 23 y la Figura 24 nuevamente corroboran el análisis realizado el cual indicia que con estos valores de tiempo de retardo generara inestabilidad, dando así validez al uso del método de CTCR.

Con los términos forzados encontrados en el Capítulo 8 se procede a simular y verificar que se mantenga la formación entre los agente, esto se evidencia en la Figura 16, donde se dan unas condiciones iniciales de posición a los agentes y ellos mediante el protocolo de consenso llegan a una posición final manteniendo una formación entre ellos.
Algo de lo que se había hablado anteriormente era acerca de no forzar el subsistema que posea $\lambda = 1$, debido a que genera una constante de velocidad en el centroide de la formación y este seguirá moviéndose hacia el infinito. Se forzara este subsistema para comprobar lo anterior mencionado, dando unos valores iguales a $\phi_3^x = 200$ y $\phi_3^y = 100$, reflejando esto en la Figura 17 donde se observa como el grupo de agentes se mantiene en formación pero no llegan a ninguna posición fija, es decir, no se encuentra estabilidad en el sistema.

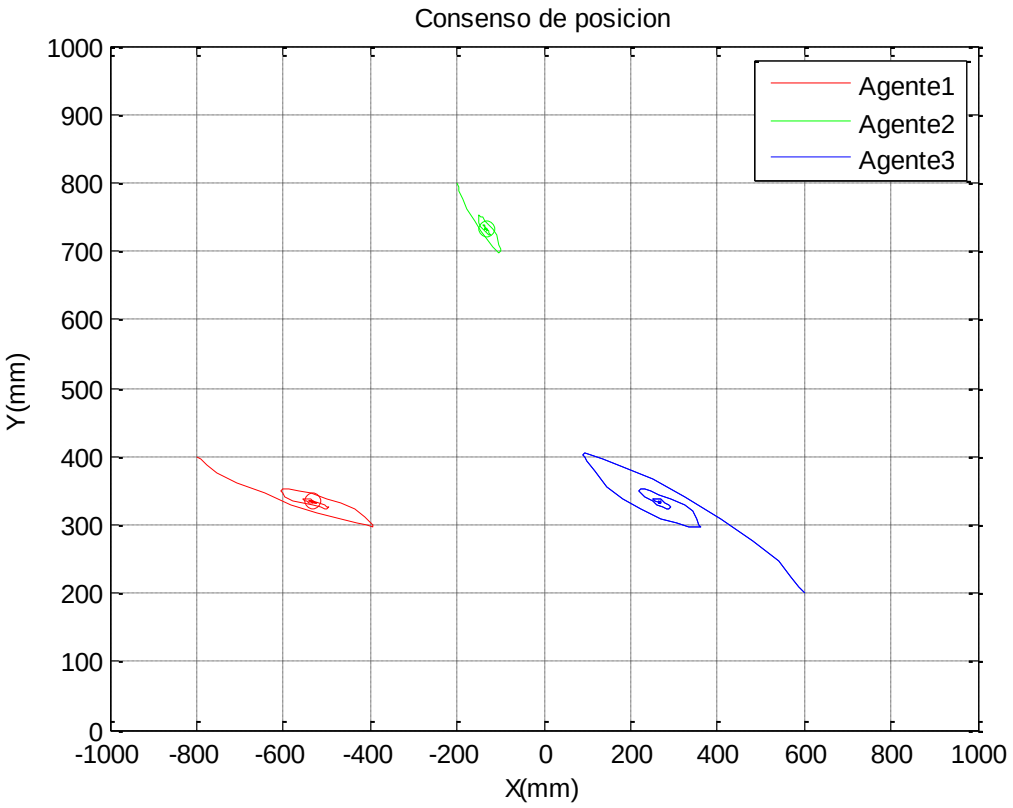


Figura 25. Formación entre agentes.

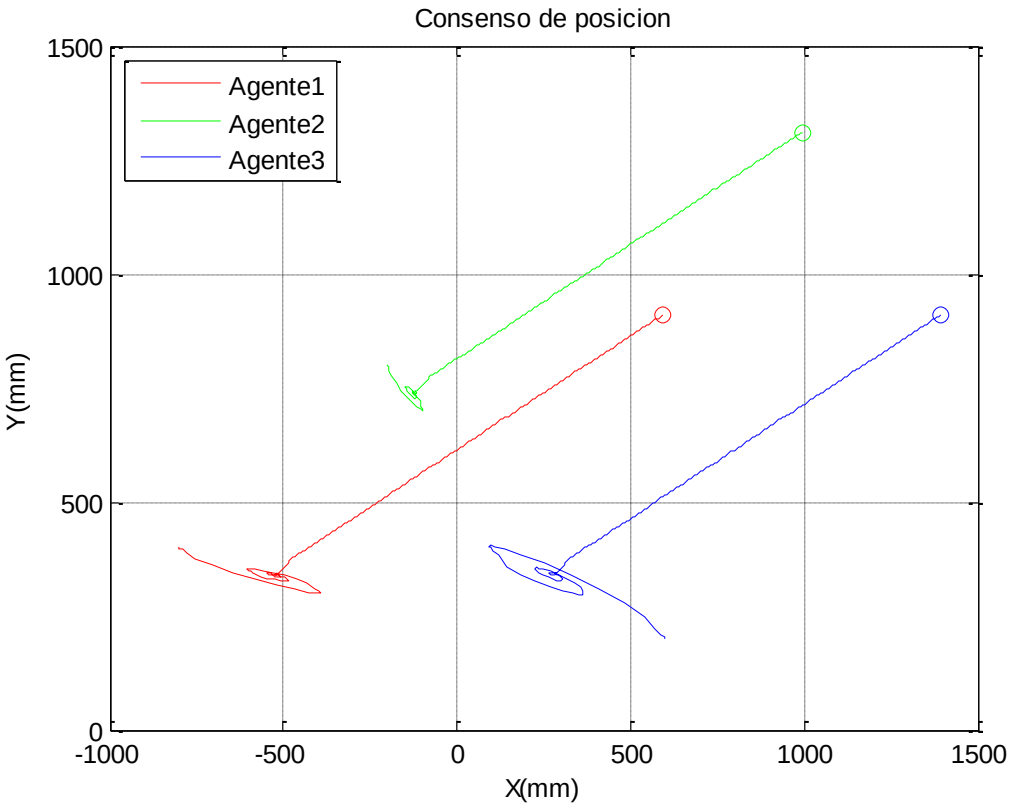


Figura 26. Formación entre agentes con $\lambda = 1$ forzado.

13. IMPLEMENTACIÓN DE HARDWARE PARA PRUEBA CON ROBOTS

El *iRobot Create* es un kit robótico de desarrollo completo que permite programar nuevos comportamientos del robot sin tener que preocuparse del montaje mecánico y el código de bajo nivel [1]. Se puede decir que es un robot móvil pre ensamblado.

La base móvil *iRobot Create* (ver figuras 27 y 28) está diseñada para maniobrar sobre superficies planas, además puede llevar cargas de hasta 2.26kg. También cuenta con un puerto de propósito general el cual permite tener control sobre el sistema y comunicarse con los actuadores y sensores. Posee también entradas y salidas digitales disponibles para adicionar elementos electrónicos propios o de terceros al sistema.

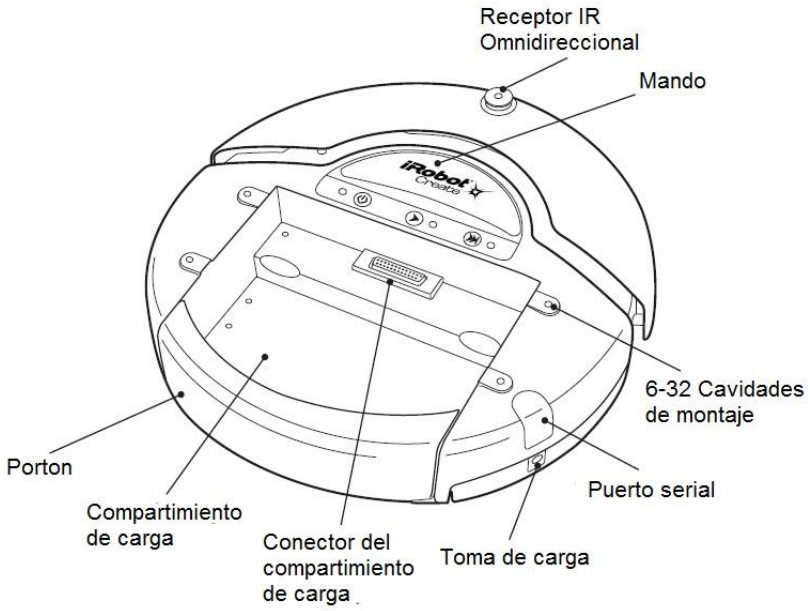


Figura 27: Irobot Create *vista frontal.*

El *Create* tiene una interfaz de comunicación llamada *Open Interface* (OI) o interfaz abierta, que permite ejecutar muchas ideas para casos prácticos y de una manera sencilla. Comandos como *Drive* (manejar) que pone en manchar la plataforma, comandos de demostración, y comandos de tocar canciones, que animan al propio desarrollo de nuevos algoritmos [1].

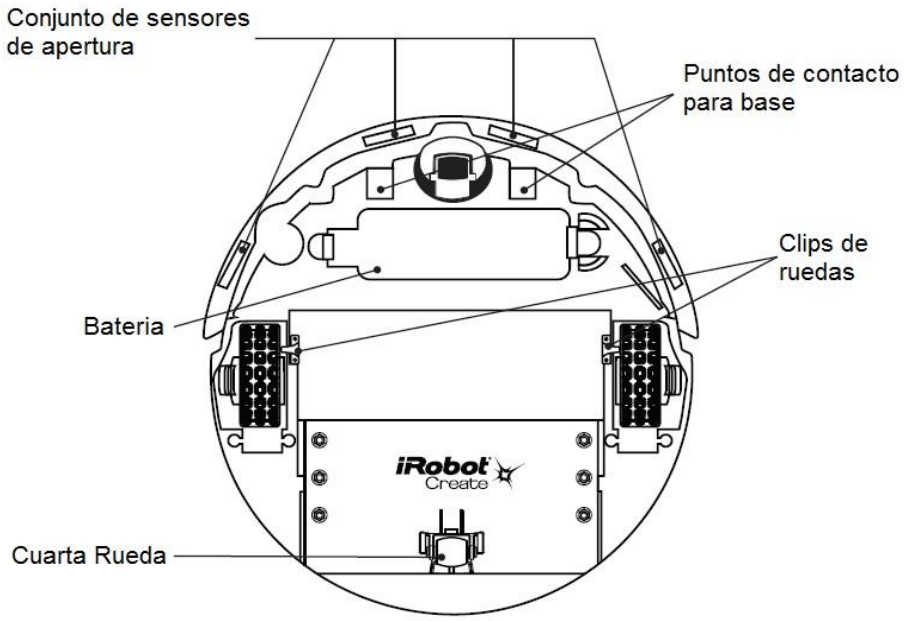


Figura 28. Irobot Create *vista inferior*

13.1 LA INTERFAZ DE COMUNICACIÓN SERIAL

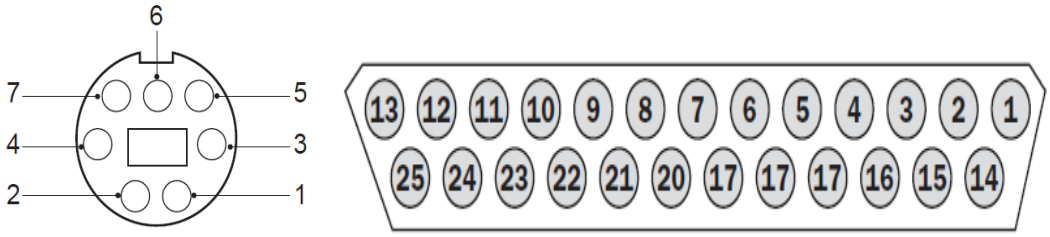
Esta plataforma cuenta con dos puertos para la comunicación serial el puerto *mini DIN* figura 5.3(a) y el *Cargo Bay Connector* que a partir de ahora se llamara Conector DB25 ver figura 5.3 (b). Estos puertos permiten una comunicación serial en ambas direcciones y TTL(0-5v)[1]. La descripción de la distribución de los pines del puerto mini DIN se puede ver en la tabla 5.1 y la del conector DB25 en la tabla 5.2. En este proyecto se usara el Conector DB25 ya que posee una cantidad de pines suficientes para amplios usos, disponibilidad del puerto en el mercado y ergonomía del mismo.

Tabla 13.1: Descripción de los pines del conector mini-DIN Fuente [12]

Pin	Nombre	Función
1	Vpwr	Voltaje de la batería, no regulado.
2	Vpwr	Voltaje de la batería, no regulado.
3	RXD	Entrada serial al robot.
4	TXD	Salida serial del robot.
5	BRC	Cambio de la tasa de transmisión.
6	GND	Referencia de voltaje.
7	GND	Referencia de voltaje.

13.1.1 Conector DB25

Esta localizado en la parte media de la compuerta de carga y contiene 25 pines etiquetados para que pueda usarse para sujetar dispositivos electrónicos y otros dispositivos periféricos como sensores adicionales, luces, o motores del *iRobot Create*[1].



(a) MiniDIN (b) DB-25

Figura 29. Distribución de pines de los conectores del Create

Tabla 13.2: Descripción de los pines del conector DB25

Pin	Nombre	Función
1	RXD	Entrada serial al robot.
2	TXD	Salida serial del robot.
3	PWR	Control de encendido.
4	AI	Entrada analógica.
5	DI1	Entrada digital 1.
6	DI3	Entrada digital 3.
7	DO1	Salida digital 1.
8	SW5	referencia regulada: 5V, 100 mA.
9	Vpwr	Voltaje de la batería, no regulada.
10	SWVpwr	Voltaje de la batería, regulado, 1.5A.
11	SWVpwr	Voltaje de la batería, regulado, 1.5A.
12	SWVpwr	Voltaje de la batería, regulado, 1.5A.
13	CHRG	Indica si el robot está cargando.
14	GND	Referencia de voltaje.
15	DD/BRC	Cambio de la tasa de transmisión.
16	GND	Referencia de voltaje.
17	DI0	Entrada digital 0.
18	DI2	Salida serial del robot.
19	DO0	Salida digital 0.
20	DO2	Salida serial del robot.
21	GND	Referencia de voltaje.
22	LSD0	Salida de 0.5A

23	LSD1	Salida de 0.5A
24	LSD2	Salida de 1.5A.
25	GND	Referencia de voltaje.

13.2. Interfaz de comandos

El interfaz trabaja en modelo consulta / retorna, donde se le envía un código del interfaz y devuelve el valor. Por ejemplo para leer la distancia en un eje se ejecuta

```
fputc(19, UART_PORT2);//Leer distancia.
d=fgetc(UART_PORT2);
```

De esta forma se interroga por el código 19 y recibe el dato a continuación. El comando serial ya depende del lenguaje que se esté trabajando.

Se realizó la construcción de dos interfaces hardware diferentes para ver cual tenía más performance para realizar la simulación.

13.3 Descripción del software para validación

Para la validación es necesario realizar tres tipos de software. Se desarrolló un software para el robot seguidor y otro para los seguidores. Así mismo se desarrolló un software en lenguaje C para el computador que monitorea en tiempo real las posiciones de cada uno de los tres robots y almacena dicha información en archivos planos para su correspondiente graficación.

Se realizaron dos tipos de interfaces para los robots en su comunicación serial. La primera se basa en microcontrolador dsPIC y el otro basado en tarjeta embebida Raspberry. Ambas interfaces implementan tecnología XBEE para comunicaciones entre ellos y con el módulo maestro que en este caso es el computador.

Se describe a continuación las especificaciones de dichas interfaces.

13.3.1 Interface dsPIC 30F4013

El proceso de cómputo debe ser capaz de comunicarse con el robot y los otros agentes y realizar operaciones numéricas con punto flotantes complejas a muy altas velocidades, para poder ejecutar los algoritmos de control cooperativo.

Las primeras dos necesidades las cumple un microcontrolador estándar, pero la clave aquí se encuentra en la tercera. Un procesador de este tipo no está diseñado para realizar operaciones numéricas complejas con números de punto flotante a altas velocidades. Por esto se hace necesario trabajar con un elemento que cuente con una ALU especialmente diseñada para este tipo de operaciones. Una opción económica y fácil de usar es el empleo de un Digital Signal Controller (Controlador Digital de Señales) o DSC. Ubicados a medio camino entre un microcontrolador y un procesador digital de señal (DSP), los DSC, particularmente la familia dsPIC de Microchip Inc., combinan la simplicidad y versatilidad de un microcontrolador y la capacidad de cómputo de un DSP.

El dsPIC30F4013, fabricado por Microchip Inc. cuenta con una CPU de 16 bits que incluye operaciones de multiplicación-acumulación en un solo ciclo de instrucciones, lo que incrementa la velocidad para realizar operaciones con datos de punto flotante. Esto es especialmente útil al hacer los cálculos que requiere el control.

Entre sus periféricos se incluyen dos transmisores-receptores asíncronos universales (UART) en hardware. Dado que el procesador tendrá que comunicarse con el robot que estará controlando y con los demás miembros del equipo, las dos unidades de comunicación se hacen necesarias.

Debido a restricciones de velocidad en el procesador por el voltaje de alimentación seleccionado, se propone el uso de un cristal externo de 10MHz como fuente para el oscilador interno. El lazo del dsPIC multiplicará esta frecuencia por cuatro, dando una velocidad de operación total de 40MHz.

Definido el procesador a emplear, se pensó en la comunicación inalámbrica. Con el fin de hacer que ésta fuera lo más transparente posible para el usuario, se decidió emplear módulos XBee básicos. Aunque tienen una gran cantidad de funciones, en su forma por defecto, estos transmisores/receptores funcionan como si simplemente reemplazaran el cable.

Ahora, dado que el módulo XBee opera a un voltaje de alimentación de 3.3 V y que éste está dentro del rango de alimentación admisible del dsPIC seleccionado, se decidió utilizar un regulador LD117AV33. Este regulador proporciona un bajo voltaje de dropout, es decir, puede mantener el voltaje de salida en el valor deseado aun cuando el voltaje de entrada caiga mucho.

El principal problema de alimentar el procesador a 3.3V es que la interfaz de comunicación serial del Create opera con niveles TTL, es decir, a 5V. Si se conectaran directamente, se podría producir un daño bien sea en el robot o en el dsPIC. Esto hace necesario el empleo de circuitos convertidores de nivel, basados en transistores de propósito general 2N3904.

Dado que el conector DB25 en la bahía de carga del robot proporciona conexiones al voltaje de la batería y a una fuente de alimentación de 5V, éstas se emplean como fuente de alimentación para todo el circuito. La fuente de 5V regulada se utilizará para el convertidor de nivel, mientras la otra se conectará al regulador que proporciona los 3.3V a los demás circuitos integrados. Además de los componentes descritos anteriormente, el diseño de la tarjeta incluye varios condensadores de tantalio para proporcionar estabilidad a las fuentes de alimentación, reduciendo el ruido de alta frecuencia que se pueda inducir en las líneas de conducción en el circuito impreso. Se incluyó también un conector de 6 pines para programar el dsPIC mediante el modo ICSP (In-Circuit Serial Programming) y un pulsador para tener la opción de reiniciar el procesador. La interconexión de todos estos componentes, incluyendo el conector DB25 que unirá la tarjeta al Create, se puede ver en el diagrama esquemático de la figura 30

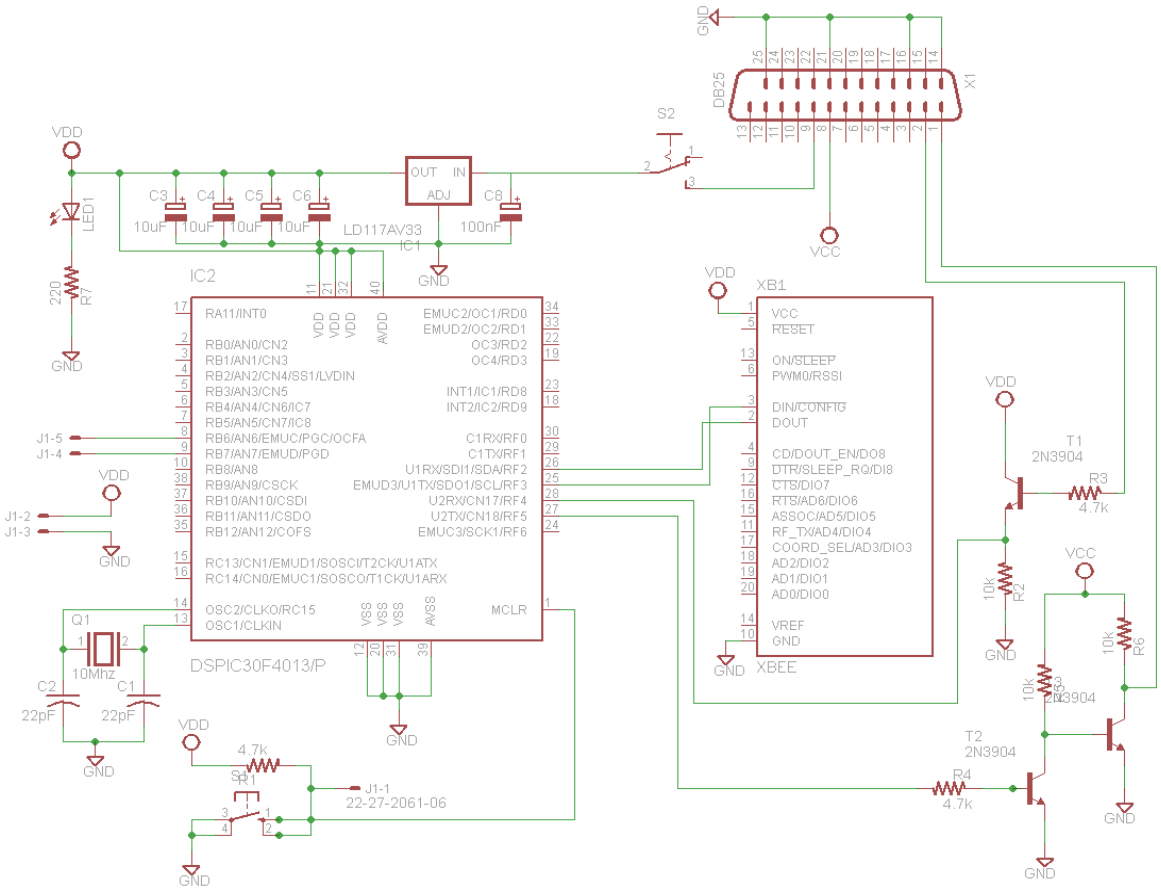


Figura 30. *Esquema de conexiones interface dsPic con robot*

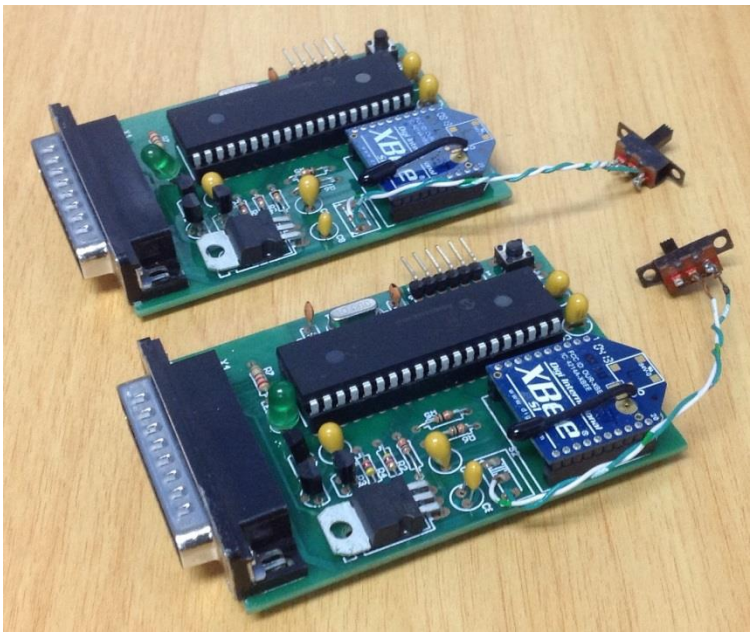


Figura 31 *Tarjeta interface dsPic con robot iCreate*

Figura 32. *Acople tarjeta interface robot iCreate*

13.3.2 Interrupciones

Debido a las necesidades del proyecto fue necesario el uso de múltiples interrupciones al mismo tiempo, lo cual ocasiona que el microcontrolador se quede ejecutando uno mientras ignora las otras, fue por esto que se optó por priorizar las interrupciones. En el caso de nuestro microcontrolador el dspic30f4013, cuenta con siete niveles de prioridades si se quiere profundizar sobre esto leer [12] las cuales nos permitieron organizar mejor nuestras interrupciones y que las atendiera por orden de relevancia.

En el caso del compilador usado el CCS compiler la función para poder priorizar las interrupciones de este tipo de microcontroladores viene desactivada por defecto, por lo que fue necesario activarla mediante la instrucción "#device nested_interrupts=true". La cual debe ser agregada justo al inicio del código ya que si no genera error el compilado. Esta instrucción nos permite asignar niveles de prioridad a las interrupciones, en nuestro acaso siete simplemente agregando la palabra high (nivel de prioridad más alta 7 para nuestro micro) al final de la declaración de la interrupción por ejemplo #INT_RDA high y las otras interrupciones quedan con nivel de prioridad cuatro. En nuestro caso esto fue suficiente ya que solo trabajamos dos interrupciones al tiempo.

A pesar de contar con una buena capacidad de cómputo en volumen de datos y velocidad de proceso, la comunicación serial basada en dsPIC no nos permitió realizar un control de los retardos dado que debe manejar simultáneamente comunicación serial con el robot para dar comandos, realizar el cálculo de posición, y manejar la comunicación serial inalámbrica XBee. El procesador no dio la capacidad deseada para la simulación de un retardo inducido.

13.3.3 Transmisión de datos

Dado que se requiere crear una topología de comunicación según el grafo representado en la figura 4 se necesitaba una forma de comunicación eficiente que permitiera que un robot dado pudiera conocer los datos de posición de los otros dos y de la misma forma los otros dos. Para esto se requiere construir tramas de comunicación que posean como mínimo el código del emisor y el código de destino, a parte de los datos propiamente dichos.

El ID como su nombre lo indica es para identificar cada agente dentro de la red, se tomara con ID 1 el agente principal el cual deberá ser seguido por el ID 2 que será el agente secundario.

El segundo dato de la trama es el DESTINO hacia que agente está destinada la información, cabe resaltar que las Xbee están en modo broadcast.

El tercer y cuarto dato es la posición en X y Y correspondiente dada en milímetros.

El quinto dato VELOCIDAD es la velocidad promedio.

13.3.4 Modo de Conexión transparente Xbee

Esta es la conexión que viene por defecto y es la más sencilla forma de configurar el módem. Básicamente todo lo que pasa por el puerto UART (DIN, pin 3), es enviado al módulo deseado, y lo que llega al módulo, es enviado devuelta por el mismo puerto UART (DOUT, pin2)[8].

Existen básicamente 4 tipos de conexión transparente. La diferencia principal radica en el número de nodos o puntos de acceso, y la forma en que éstos interactúan entre sí.

En este caso se explica el modo Broadcast que fue el seleccionado para el proyecto.

Para usar la Xbee en modo Broadcast fue necesario usar el programa X-CTU disponible en la web del fabricante y cambiar los valores predeterminados de las mismas.

13.3.5 Broadcast

Esta configuración permite el envío de información desde un nodo a varios nodos en una misma red. La información recibida es la misma para todos los nodos. Para configurar los módulos, es necesario ajustarlos con la dirección de Broadcast. Cualquier módulo que reciba un paquete con una dirección de destino de Broadcast será aceptado[8].

La dirección de Broadcast es:

DL=0x0000FFFF

DH=0x00000000

Esta dirección debe ser configurada en todos los nodos de la red, ya sea que estén en direccionamiento de 16 o 64 bits. Así se debe ingresar ATDH0 y ATDL0000FFFF en todos los módulos para que el modo broadcast esté habilitado. La figura 8.3 muestra una red de Broadcast:

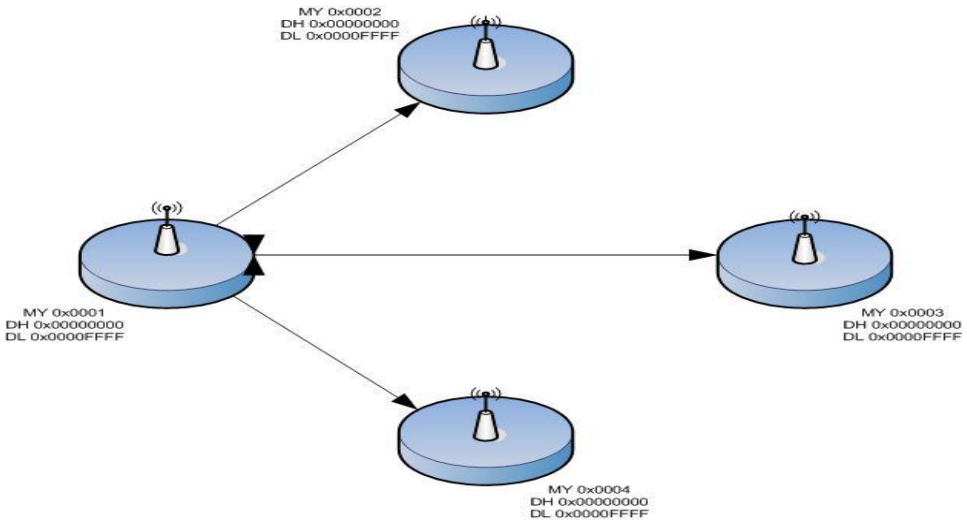


Figura 33. Red de Broadcast.

Se observa en la anterior la configuración de Broadcast. Si se envía algún dato por el módulo 0x0001, la información enviada será recibida por igual en el resto de los módulos (0x0002, 0x0003 y 0x0004). Del mismo modo si se envía algún dato por otro módulo, por ejemplo por el 0x0004, este dato le llegará al resto, es decir, al 0x0001, 0x0002 y al 0x0003. Cabe mencionar que este tipo de red o de envío de datos, no entrega respuesta de recibo o ACK, por lo que no es posible saber si el paquete fue entregado correctamente o si es que llegó.

Si se ajusta la dirección PAN ID del módulo como ID=0xFFFF, se produce Broadcast a todas las redes PAN. Esto es, los datos son transmitidos a las distintas redes PAN, pero no se confirma la entrega de éstos (no se recibe ACK). Si se ingresa ID=0xFFFF y además DL=0xFFFF se realiza doble broadcast, es decir, además de transmitirse los datos a todas las redes PAN, el mensaje es transmitido a todos los módulos de cada una de ellas. Si se ingresa ID=0xFFFF y DL=0xAAAA (dirección arbitraria), los datos son transmitidos a todos los módulos que posean la dirección AAAA, pero que no necesariamente se encuentren en la misma red PAN[15].

Las tareas encomendadas al microcontrolador comenzaron a rebasar su capacidad y a pesar de usar interrupciones cuando se construyó la topología expuesta anteriormente las tramas comenzaron allegar truncadas y el retardo en el proceso hacía que el robot corrigiera tarde ocasionando que solo funcionaba con bajas velocidades de movimiento del robot. No se podía entonces

13.4 Interfaz Rsapberry PI

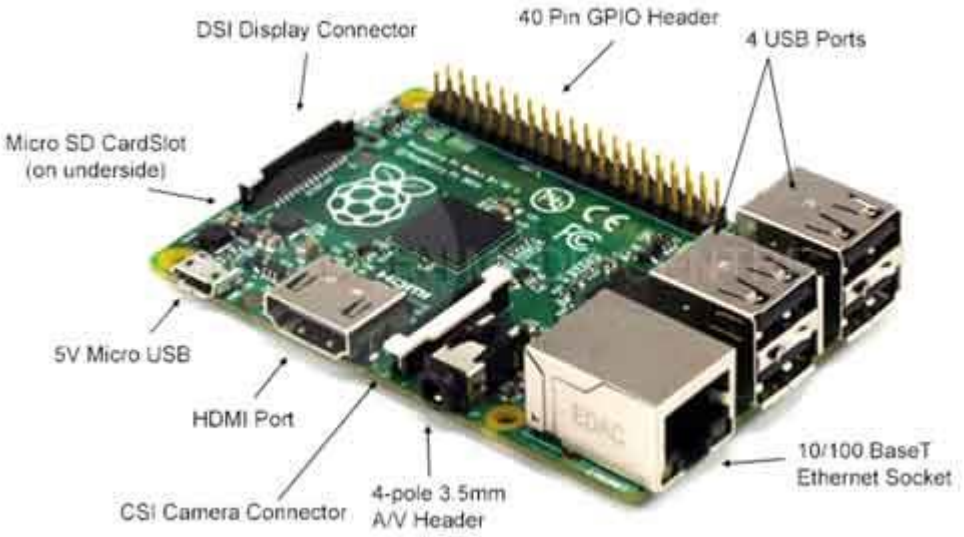


Figura 34. Tarjeta Raspberry PI 3

La Raspberry es un computador de placa reducida usada educativamente para proyectos de ciencia y tecnología, así como para pequeños proyectos de automatización. La placa, que antes era más pequeña que una tarjeta de crédito tiene varios puertos y entradas, dos USB, uno de Ethernet y salida HDMI. Estos puertos permiten conectar el miniordenador a otros dispositivos, teclados, ratones y pantallas.
Además de un ordenador Raspberry incorpora funciones de electrónica como pines GPIO (General Purpose Input/Output), y de comunicación como UART (Universal Asynchronous Receiver-Transmitter), y SPI (Serial Peripheral Interface), I²C (Inter-Integrated Circuit).

Estas funciones hacen que pueda ser empleado en proyectos de electrónica y robótica interactuando con sensores (temperatura, luz, aceleración…) y actuadores (motores, servos, relés, etc).

El puerto digital GPIO nos permite la comunicación serial con los robots icreate. Sin embargo para este proyecto se usó la comunicación por la bahía serial conector PS2.

La descripción de dicho puerto se ve en la figura 35

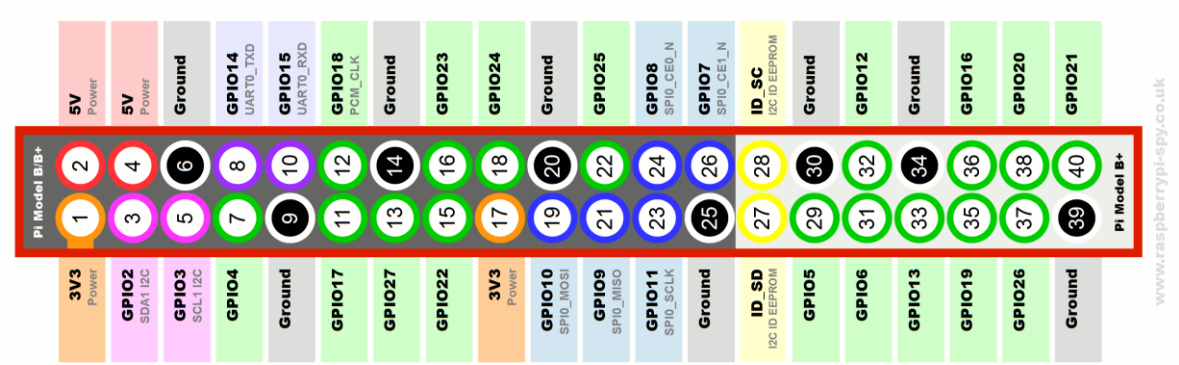


Figura 35. *Puerto GPIO de la Raspberry PI 3*

14. DESARROLLO DEL SOFTWARE EN LA PLATAFORMA ICREATE

El programa desarrollado en las tarjetas Raspberry será descrito a continuación. Es de tener en cuenta que se utilizaron dos puertos USB uno para la comunicación serial XBee y el otro para la comunicación serial con el robot, a través de su puerto con conector PS2

A diferencia del código desarrollado para la tarjeta ensamblada con microprocesador dsPIC donde había un robot líder y los otros lo seguían, en esta implementación todos los robots tienen el mismo código y todos transmiten su telemetría por xbee en modo broadcast. El código fue implementado en Python el cual es un lenguaje interpretado y no requiere compilación previa.

Con el fin de independizar los procesos de control del robot, comunicación de la raspberry con el mismo y xbee, se utilizó programación concurrente en forma de multihilo, lo que da un performance mcho mejor de la tarjeta y nos permite incorporar retardos simulados que son manejados casi con la exactitud requerida.

Función lee_xbee: lee la cadena transmitida por otros robots donde se lee su código de identificación y la posición y velocidad de cada uno de los otros robots.

Función escribirXbee: arma la cadena de la forma
ser.write(cadena.format(robot,int(x),int(y),int(vx),int(vy))), donde envía su osición y velocidad en cada eje.

Función hilo1: realiza el armado de la trama para manejar los movimientos de cada uno de los motores según el cálculo realizado en hilo2

Función hilo2: realiza el cálculo de ángulo y distancia a la que debe dirigirse el robot según el protocolo de consenso descrito en el capítulo 4

Función proceso: Realiza el cálculo del protocolo de consenso como tal. Básicamente con las siguientes dos instrucciones implementa el cálculo de velocidad y distancia que debe realizar el robot en el siguiente ciclo del programa

```
x = x + (dx * cos(ang0)) - (dy * sin(ang0))
y = y + (dx * sin(ang0)) + (dy * cos(ang0))
```

Dichas instrucciones corresponden al cálculo de protocolo descrito anteriormente.

15. CONCLUSIONES.

El uso del protocolo CTCR propósito principal de este trabajo en una formación de tres robots no holonómicos arroja un resultado satisfactorio desde la posibilidad de implementación práctica teniendo en cuenta retardos fijos que en este caso fueron simulados para poder tener un control en la prueba donde se validaran los resultados experimentales obtenidos en Matlab por medio de las hipersuperficies kernel básicas y de descendencia.

El protocolo demostró alta exactitud dado que en la simulación y en la implementación real con robots se vio que si estaba en los límites de tiempo de los dos retardos el sistema permanecía estable, logrando consenso en posición de los tres robots.

Es importante anotar que si el sistema estaba en una zona de no estabilidad por cualquiera de los dos retardos, o por los dos, al aumentar o disminuir dichos tiempos se podía llevar a una zona de estabilidad. Esta conclusión es muy importante ya que si un sistema es inestable sin presencia de tiempos de retardos, este puede llegar a ser estable en presencia de ellos, por lo que en la práctica es eficiente ya que no tendría que ser alterado el sistema, sino que se puede hacer uso de un retardo programado para solucionar los problemas de estabilidad

16. BIBLIOGRAFÍA.

[1] Sergey Dashkovskiy and Lars Naujok, “Lyapunov-Razumikhin and Lyapunov-Krasovskii theorems for interconnected ISS time-delay systems," Proceedings of the 19th International Symposium on Mathematical Theory of Networks and Systems, 2010.

[2] Qingdong Li et al, “Containment analysis and design for general linear multi-agent systems with time-varying delays," Neurocomputing 173, pp. 2062–2068, 2016.

[3] Zhaoxia Peng et al, Group Consensus Control for Heterogeneous Multi-Agent Systems with Fixed and Switching Topologies, International Journal of Control , 2015

[4] Derong Liu et al, “Distributed control algorithm for bipartite consensus of the nonlinear time-delayed multi-agent systems with neural networks," Neurocomputing 174, pp. 928–936, 2016.

[5] Ming Xin and Jianan Wang, “Distributed optimal cooperative tracking control of multiple autonomous robots," Robotics and Autonomous Systems 60, pp. 572–583, 2012.

[6] Da Zheng, Zhengyun Ren and Jian-an Fang, “Stability analysis of multiple time-delayed system," ISA Transactions 47, pp. 439–447, 2008.

[7] Shichun Yang et al, “Adaptive distributed formation control for multiple nonholonomic wheeled mobile robots," Neurocomputing 173, pp. 485–1494, 2016.

[8] López Moreno, Hermes. Diseño de un módulo de extensión para aplicaciones de control cooperativo con la plataforma irobot créate (tesis de pregrado). Universidad Santo Tomás, Colombia, 2014

[9] OGATA, Katsuhiko. Ingeniería de control moderna. Quinta edición. Traducido por Sebastián Dormido Canto. PEARSON EDUCACIÓN, S.A., ISBN:978-84-8322-660-5, 2010

[10] Oriolo et al, “WMR Control Via Dynamic Feedback Linearization:
Design, Implementation, and Experimental Validation," IEEE TRANSACTIONS ON CONTROL SYSTEMS TECHNOLOGY, vol. 10, no. 6,pp. 835-852, 2002.

[11] Cepeda Gomes, Rudy. Exact and Exhaustive Stability Analysis of Linear Consensus Protocols with Time-Delay (tesis doctoral). University of Connecticut, United States. 2012

[12] N. Olgac and R. Sipahi, An Exact Method for the stability Analysis of Time-Delayed Linear Time invariant Systems," IEEE Transactions on Automatic Control, vol. 47, no. 5, pp. 793-797, 2002.

[13] H. Fazelinia, R. Sipahi, and N. Olgac, Stability robustness analysis of multiple timedelayed systems using building block concept," IEEE Transactions on Automatic Control, vol. 52, no. 5, pp. 799{810, 2007.

[14] Rudy Cepeda-Gomez, Nejat Olgac. Exhaustive Stability analysis in a consensus system with time delay and irregular topologies, International Journal of Control, vol. 84, no. 4, pp. 746-757, April 2011.

[15] R. Olfati-Saber and R. Murray, “Consensus problems in networks of agents with switching topology and time-delays," IEEE Transactions on Automatic Control, vol. 49, no. 9,pp. 1520-1533, 2004.

[16] R. Cepeda-Gomez and N. Olgac, “Formation Control Based on a Consensus Protocol under Directed Communications with Two Time Delays," IEEE Transactions on Automatic Control, Submitted, 2012.

[17] R. D. Schaefer, An Introduction to Nonassociative Algebras. Dover, 1996.

[18] De la Fuente Aparicio, María Jesús. Aplicaciones de las redes de neuronas en supervisión, diagnosis y control de procesos. Equinoccio, ISBN: 9802371904, 1999

[19] A. F. Ergenc, N. Olgac, and H. Fazelinia. Extended Kronecker summation for cluster treatment of LTI systems with multiple delays," SIAM Journal on Control and Optimization, Volume 47, 2014

[20] V. Putranti, Z. H. Ismail, T. Namerikawa, "Robust-formation control of multi-autonomous underwater vehicles based on consensus algorithm", 2016 IEEE Conference on Control Applications (CCA), pp. 1250-1255, Sept 2016

[21] B. Arbanas, T. Petrovic, S. Bogdan, "Consensus protocol for underwater multi-robot system using scheduled acoustic communication", *To appear in the proceedings OCEANS 2018 IEEE/MTS*, 2018.

[22] V. Putranti, Z. H. Ismail, T. Namerikawa, "Robust-formation control of multi-autonomous underwater vehicles based on consensus algorithm", *2016 IEEE Conference on Control Applications (CCA)*, pp. 1250-1255, Sept 2016.

[23] Arévalo Illescas, Bryan Michael. Diseño e implementación de un controlador SMC-Delay para seguimiento de trayectoria de una formación de robots móviles con retardo de entrada. : EPN.Escuela Politécnica Nacional de Quito, 2018

[24] IngbinGao, NejatOlgac. Dixon Resultant for Cluster Treatment of LTI Systems with Multiple Delays, IFAC-PapersOnLine, Volume 48, Issue 12, Pages 21-26, 2015

[25] Mihir R. Ghooi, Yogesh V. Hote. Design of PI controller for position control system using Kronecker summation approach: Experimental validation. International Conference on Power, Instrumentation, Control and Computing (PICC), 2018